FM-7
セブン

F-BASIC 解析マニュアル
フェーズⅠ 基礎編

中村英都 著

SHUWA SYSTEM TRADING CO.,LTD.

FM-7
F-BASIC
ANALYTICAL MANUAL

SHUWA SYSTEM TRADING CO.,LTD.

まえがき

富士通のパーソナルコンピュータFujitsu Micro 7（FM7）は，以前に発表されたFM8の実績をふまえた上で改良とコストダウンをはかり，12万円台という低価格で発表され，着実に台数を伸ばしています．

このFM7は，ホビーユースを意識したマシンであるにもかかわらず，FM8からうけついだ大きな可能性を秘めています．FM7は究極の8bitマイクロプロセッサ68B09を搭載，しかもクロック周波数2MHzの高速動作であること等，数多くの魅力をもったパーソナルコンピュータといえます．

しかし，この魅力もFM7を単なるBASICマシンとして使用していたのでは半減してしまいます．そこでマシン語を使用することになるわけですが，特にハードウェアと密接に関わっている部分のプログラミングは繁雑な点が多くなります．幸い，FM7にはハードウェアとのI／OをつかさどるBIOSが搭載され，その仕様も公開されており，また，BASIC内部のシステムサブルーチンには有用なものも多く含まれ，これらを利用することにより，プログラミングは容易なものとなります．

本書では，BIOSおよびDisplay Sub Systemの使用法を豊富なサンプルプログラムをつけ加えて説明し，その理解の一助とすると同時に，BASIC内部の有用なシステムサブルーチンをピックアップして解説しています．このように本書は，FM7の魅力を十二分に引き出し，その持てる機能を思うままに活用できるようにするため，筆者らが独自に解析したFM7のファームウェアの内容をまとめ上げたものです．

読者の方々が，本書をマシン語プログラミングの手引書として利用していただければ，本望と思っています．

本書の執筆・出版に当って，多くの方々にお世話になりましたことを，深く感謝いたします．

1983年10月

中村　英都

参考文献

1. 「ＦＭ７　ユーザーズマニュアル　システム解説」，富士通
「ＦＭ７　ユーザーズマニュアル　システム仕様」，富士通
「ＦＭ７　Ｆ－ＢＡＳＩＣ文法書」，富士通

2. 「マイコンピュータ　ＮＯ２」，ＣＱ出版社，ｐ１０３～１１５

3. 「ＦＭ８　活用研究」，工学社
「ＦＭ７／８　活用研究」，工学社

4. 「ＰＣ８００１　マシン語活用ハンドブック　初級編」
「ＰＣ８００１　マシン語活用ハンドブック　中級編」
「ＰＣ８００１　ＳＯＵＲＣＥ　ＰＲＯＧＲＡＭ　ＬＩＳＴＩＮＧＳ」
「ＰＣ８８０１　N_{88}ＢＡＳＩＣ解析マニュアル」
以上　秀和システムトレーディング株式会社

解析に使用したシステム

●ＦＭ７本体　ＭＢ２５０１０　Ｓ／Ｎ　６８２１１０６２０
　Ｆ－ＢＡＳＩＣ　Ｖ０３．００　'８２．０９．２０
　（＄ＦＢＡ５～＄ＦＢＡＥに格納）
　ＢＩＯＳ　Ｖ１Ｌ１　'８２．０９．０１
　（＄Ｆ１７６～＄Ｆ１７Ｃに格納）

●ＭＢ２７６０７，ＭＢ２７６０８　薄型ミニフロッピィディスクユニット

●ＭＢ２７４０５　シリアルドットプリンタ

●Ｆ－ＢＡＳＩＣ　Ｖ３．０　システム・ディスク
　'８２．１０．０１（＄６ＥＡ３～＄６ＥＡ８に格納）

●富士通「ＦＭ７用アブソリュートアセンブラ」

●富士通「ＦＭ７用デバッガ／逆アセンブラ」

〈本文内の略記法について〉

A レジスタ，AccA，A　＝アキュームレータA
B レジスタ，AccB，B　＝アキュームレータB
D レジスタ，AccD，D　＝倍長アキュームレータD
CCR，CC レジスタ　＝コンディションコードレジスタ
DPR，DP レジスタ　＝ダイレクトページレジスタ
EA　＝実効アドレス
IX，X，X レジスタ　＝インデックスレジスタX
IY，Y，Y レジスタ　＝インデックスレジスタY
PC，PCR　＝プログラムカウンタ
$　＝16進
S，SP　＝ハードウェアスタックポインタ
U，US　＝ユ--ザースタックポインタ
n　＝10進数1ケタ
h　＝16進数1ケタ
←　＝代入
(　)　＝カッコ内の値をアドレスとしたとき，そのアドレスの内容
FAC　＝フローティング・アキュームレータ
∧A　＝コントロールA
reg　＝レジスタ

Ⅰ　バイオス解説……………………………………………………1

BASIC input output system

III　システム・サブルーチン　1……………115

System Subroutines 1

1. BASIC Input Output System

1―1　ＢＩＯＳの概略

Ｆ―ＢＡＳＩＣでは，Ｉ／Ｏ機器への入出力処理のうち，割り込み処理と，ＲＳ２３２Ｃ関連機器への入出力を除いた入出力処理を，ＢＩＯＳ（Basic Input Output System）と呼ばれるプログラムモジュールを介して行っています．

このことは，ユーザーがＩ／Ｏ機器への入出力を行なう際，ＢＩＯＳを使用することにより，比較的簡単に各Ｉ／Ｏ機器への入出力を行えるという利点をもたらします．

このＢＩＯＳは，以下のＩ／Ｏ機器をサポートしています．

(1)　アナログ入力ポート
(2)　オーディオカセット
(3)　ブザー
(4)　漢字ＲＯＭ
(5)　プリンタ
(6)　ディスク
(7)　ディスプレイ・サブシステム（ディスプレイ・キーボード）
(8)　バブルカセット

これらの機器への入出力は，ＢＩＯＳ内のドライバルーチンをサブルーチンコールすることにより行います．このとき，ＢＩＯＳの入口（エントリ・アドレス）は，１ヶ所に集中されており各リクエストの指定やパラメータの受渡し等はすべてＲＣＢ（Request Control Block）とよばれる領域を介して行います．

ＲＣＢは通常ＲＡＭ上の８バイトを必要としますがリクエストによって必要なバイト数に違いがあります．

簡単なＢＩＯＳの使用手順をフローチャートに示します．パラメータの内容等はリクエストによって違いがあるので，各リクエストの解説を御参照下さい．

実際にＢＩＯＳをサブルーチンコールする方法には４つの方法があります．

(1)　ＪＳＲ　［＄ＦＢＦＡ］
(2)　ＪＳＲ　＄ＤＥ　(ただしＤＰレジスタ＝＄００のとき)
(3)　ＪＳＲ　＄Ｆ１７Ｄ
(4)　ＢＩＯＳのリクエスト判断・分岐のルーチンを介さずに直接各リクエストのジャンプ先をコールする．ただしこのときＤＰレジスタ＝＄ＦＤでなければならず，レジスタ内容も保障されない．

注) このうち(4)は使用しない方が望ましく，また(3)もＦＭ８との互換性を欠くのでその点に注意する必要があります．ちなみにＦ－ＢＡＳＩＣ

内部では(2)の方法が使用されていますが，(1)の方法を用いるのが最適となります．

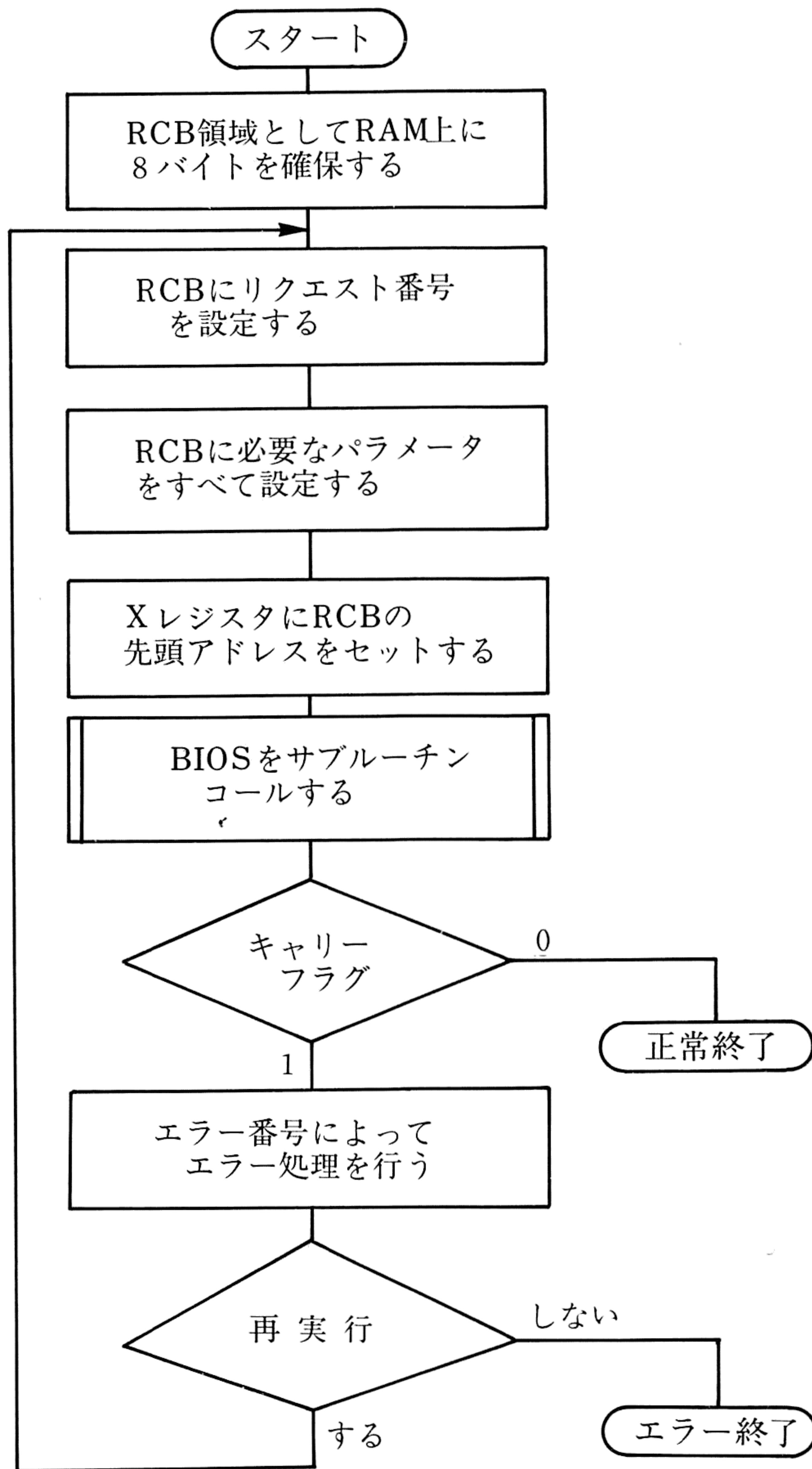
また，ＢＩＯＳはＣＣレジスタ（Ｅ，Ｆ，Ｉは保存）を除いてレジスタ内容を保存します．

ＢＩＯＳはＩ／Ｏを扱っているためＩ／Ｏ機器にエラーが生じる場合があります．このときＢＩＯＳはエラー番号を（ＲＣＢ＋１）にセットすることにより，ユーザに対してエラーが生じたことを伝えます．同時にキャリーフラグもセットするのでキャリーフラグを見ることによってもエラーの有無を知ることもできます．しかしＢＩＯＳはエラーに対してユーザーへ報告するだけでエラー回復処理やリトライなどの処理は一切行わないので，エラーが生じた場合にはユーザー側で適切な処置を施す必要があります．

エラーが生じず正常にリクエストを終了したときには（ＲＣＢ＋１）に＄００がセットされ，キャリーフラグはクリアされます．またリクエストによってはエラーが生じないものもあり，この場合でも＄００がセットされます．

また，ＢＩＯＳはＢＩＯＳのワークエリアとして＄ＦＦＥ０～＄ＦＦＥ８を使用しているので，ユーザーはここを変更してはいけません．このワークエリアはディスクの入出力とサブシステムの入出力において使われています．

＄ＦＦＥ０　最後にアクセスしたディスクドライブの番号（０～３）
＄ＦＦＥ１　ドライブ０のヘッドのあるトラック番号（０～３９）
＄ＦＦＥ２　ドライブ１のヘッドのあるトラック番号（０～３９）
＄ＦＦＥ３　ドライブ２のヘッドのあるトラック番号（０～３９）
＄ＦＦＥ４　ドライブ３のヘッドのあるトラック番号（０～３９）
＄ＦＦＥ５　bit 7がディスクのモータの状態を示しています．bit 7が１であるとモータが回転していることを示しています．
＄ＦＦＥ６～＄ＦＦＥ７　サブＣＰＵとのコマンドのやりとりで使用します．
＄ＦＦＥ８　ドライブ選択時に使用します．ディスク関係のリクエストにおいて，選択するディスクドライブの番号（０～３）を示します．

スタート
RCB領域としてRAM上に
8バイトを確保する
RCBにリクエスト番号
を設定する
RCBに必要なパラメータ
をすべて設定する
Xレジスタにrcbの
先頭アドレスをセットする
BIOSをサブルーチン
コールする
キャリー
フラグ
0
正常終了
1
エラー番号によって
エラー処理を行う
再 実 行
しない
エラー終了
する

1―2　各リクエスト解説

ＢＩＯＳには，全部で２８個（＄００～＄１Ｂ）のリクエストがありますが，そのうち４つは無効です．リクエストの指定はＲＣＢの１バイト目（（ＲＣＢ＋０））にリクエスト番号をセットすることにより行います．ＢＩＯＳはこの番号を読み取り各処理を実行します（リクエスト番号でない番号を指定した場合にはＢＩＯＳエラーとなります）．

ＢＩＯＳリクエスト番号表

１６進	10進	*** ラベル名	１６進	10進	ラベル名
＄００	0	ＡＮＡＬＧＰ	＄０Ｅ	１４	ＬＰＯＵＴ
＄０１	1	ＭＯＴＯＲ	＄０Ｆ	１５	ＨＤＣＯＰＹ
＄０２	2	ＣＴＢＷＲＴ	＄１０	１６	ＳＵＢＯＵＴ
＄０３	3	ＣＴＢＲＥＤ	＄１１	１７	ＳＵＢＩＮ
＄０４	4	* ＩＮＴＢＢＬ	＄１２	１８	ＩＮＰＵＴ
＄０５	5	ＳＣＲＥＥＮ	＄１３	１９	ＩＮＰＵＴＣ
＄０６	6	* ＷＲＴＢＢＬ	＄１４	２０	ＯＵＴＰＵＴ
＄０７	7	* ＲＥＤＢＢＬ	＄１５	２１	ＫＥＹＩＮ
＄０８	8	ＲＥＳＴＯＲ	＄１６	２２	ＫＡＮＪＩＲ
＄０９	9	ＤＷＲＩＴＥ	＄１７	２３	ＬＰＣＨＫ
＄０Ａ	１０	ＤＲＥＡＤ	＄１８	２４	ＢＩＩＮＩＴ
＄０Ｂ	１１	**	＄１９	２５	**
＄０Ｃ	１２	ＢＥＥＰＯＮ	＄１Ａ	２６	**
＄０Ｄ	１３	ＢＥＥＰＯＦ	＄１Ｂ	２７	**

*：バブル関係のリクエスト．バブルについてはＦ－ＢＡＳＩＣでもサポート対象から除外されたので，これらのリクエストに関しては解説しません．

**：無効となるリクエスト．ＦＭ８では音声合成のリクエストでありこれらのリクエストは何もせずに戻ります（エラーも生じません）．

***：混乱をさけるためラベル名は「ユーザーズマニュアルシステム仕様」と同一のものを使用しています．

<各項目の見方>

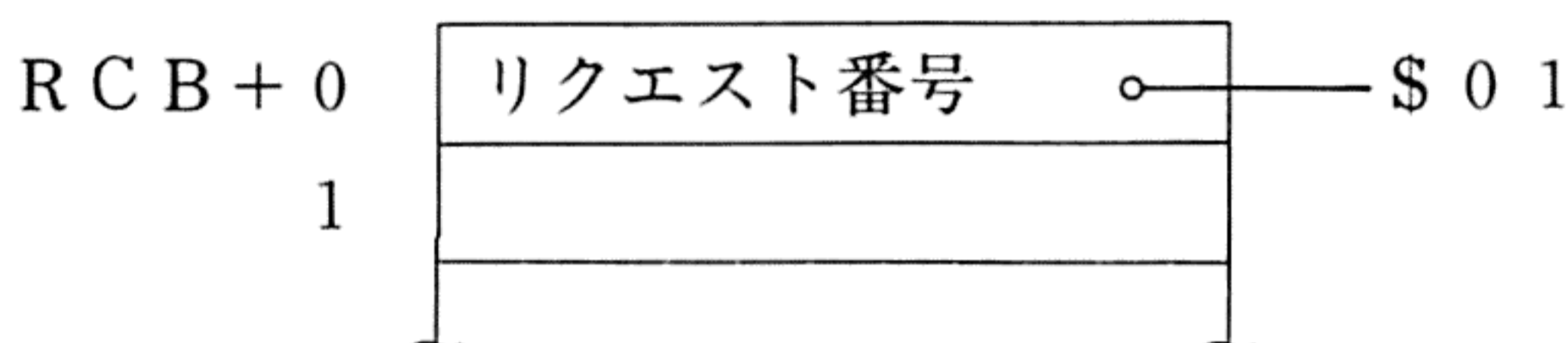
各項目は用途別に分類してあります．

機　能　各ルーチンの機能を簡単に示します．

パラメータ　各ルーチンを呼び出すにあたって設定すべきパラメータについて示します．ＢＩＯＳについてはＲＣＢのセットが中心となるので，ＲＣＢについては簡単な図を使用しています．

例）　Ｘレジスタ←ＲＣＢ先頭アドレス

インデックスレジスタＸにＲＣＢの先頭アドレスをセットすることを示します．

ＲＣＢ＋０	リクエスト番号	＄０１
１		

ＲＣＢ＋０番地にリクエスト番号をセットすることを示します．

さらに，このときのリクエスト番号は＄０１であることを示します．

復帰情報　各ルーチンを呼び出したあと，ルーチンの側から返される情報について示します．

解　説　各ルーチンの詳しい解説，注意事項，予備知識等を示します．

サンプル　いくつかのルーチンについてサンプルプログラムを付けました．サンプルプログラムは一部を除いてマシン語で書かれており，適宜実行例を付してあります．

なおアセンブルは富士通㈱のアブソリュートアセンブラを使用したものです．この際，ＦＣＢ，ＦＣＣ，ＦＤＢなどの擬似命令ではすべてのデータを出力しない指定（ＮＯＧＥＮ）がしてありますので，モニタなどでオブジェクトを打ちこむ方は注意して下さい．（詳しくは付録を参照）

また，マシン語プログラムは特に示してあるものを除いてポジション・インディペンデント（位置独立）です．ただし，実行に当って，ＤＰレジスタは０であることが必要です．

BIOS：BIINIT　－ＢＩＯＳイニシャライズ－

機　能　ＢＩＯＳのイニシャライズを行う.

パラメータ　　Ｘレジスタ←ＲＣＢ先頭アドレス

ＲＣＢ＋０	リクエスト番号	＄１８
１		

このリクエストはＲＣＢ領域としては，２バイトしか必要としません.

解　説　このリクエストは具体的には，スピーカーを動作可能にし，プリンタに対してダミーのデータ（＄００）を送出するだけです.

Ｆ－ＢＡＳＩＣでは，リセット時及びアボート時に，このリクエストを実行しているので，ユーザーがＢＩＯＳの他のリクエストに先立って，このリクエストを呼ぶ必要はありません.

BIOS：ANALGP　－アナログポート入力－

機　能　アナログポートから，Ａ／Ｄ変換されたデータを読み取る.

パラメータ　　Ｘレジスタ←ＲＣＢ先頭アドレス

ＲＣＢ＋０	リクエスト番号	＄００
１		
２	チャンネル番号	０～３
３	電圧レンジ	’Ｈ’（＄４８）……０～2.5 Ｖ ＄４８以外………０～0.625 Ｖ
４		

このリクエストはＲＣＢ領域としては，５バイトしか必要としません.

復帰情報　（ＲＣＢ＋４）：符号なし８ビットのＡ／Ｄ変換データ

解　説　このリクエストは，ＦＭ７にＦＭ８フルコンパチブルのＡ／Ｄ変換基板（数社から発売されている）を実装していないと意味をもちません.

基板を実装している場合には，Ｆ－ＢＡＳＩＣのＡＮＰＯＲＴ関数も，正常に動作します.

このリクエストでは，エラーは生じません.

サンプル アナログポートから，スペースキーを押すごとにデータを入力し表示します．基板を実装していない時には常に２５５が返されます．

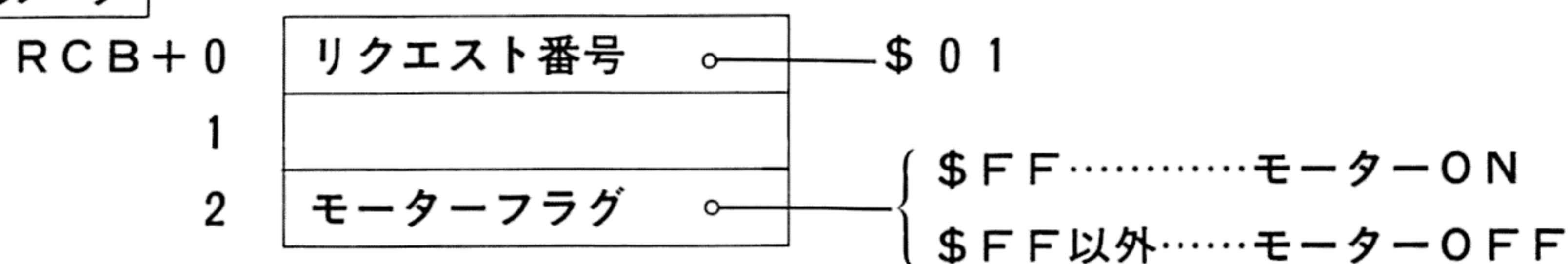
```
PAGE   001 (         ,          )

01000                       *
01010                       * GET A/D CONVERTED DATA
01020                       *                 (BIOS:ANALGP)
01030                               OPT     NOO,M,NOS,P=255,NOG
01040            9BDB       MESSAG  EQU     $9BDB   output from (X+1) till 0
01050            DB54       KEYIN   EQU     $DB54   key input with wait
01060            B615       D1OOUT  EQU     $B615   output Dreg. in decimal
01070            D0BE       OUT     EQU     $D0BE   one character out
01080            FBFA       BIOS    EQU     $FBFA   BIOS (extend-indirect)
01090                       *
01100    5000                       ORG     $5000
01110            5000       START   EQU     *
01111                       *
01115                       * set channel no. (0..3)
01116                       *
01120    5000 30 8C 54              LEAX    <MES1-1,PCR message output
01130    5003 BD 9BDB               JSR     MESSAG
01140    5006 BD DB54       GETCHA  JSR     KEYIN   get keyinput
01150    5009 81 30                 CMPA    #'0     keyinput must be '0'..'3'
01160    500B 25 F9    5006         BLO     GETCHA
01170    500D 81 33                 CMPA    #'3
01180    500F 22 F5    5006         BHI     GETCHA
01190    5011 BD D0BE               JSR     OUT     echo back
01200    5014 84 03                 ANDA    #03     channel no. 0..3
01210    5016 B7 508D               STA     CHANEL  set cannnel no.
01211                       *
01215                       * set voltage range ('H'or'L')
01216                       *
01220    5019 30 8C 47              LEAX    <MES2-1,PCR message output
01230    501C BD 9BDB               JSR     MESSAG
01240    501F BD DB54       GETRAN  JSR     KEYIN   get keyinput
01250    5022 81 48                 CMPA    #'H     keyinput must be 'H'or'L'
01260    5024 27 04    502A         BEQ     R01
01270    5026 81 4C                 CMPA    #'L
01280    5028 26 F5    501F         BNE     GETRAN
01290    502A BD D0BE       R01     JSR     OUT     echo back
01300    502D B7 508E               STA     RANGE   set range
01309                       *
01310                       * get A/D converted data
01311                       *
01320    5030 30 8C 55      DATAIN  LEAX    <RCB,PCR load rcb address
01330    5033 86 00                 LDA     #$00    bios request ANALGP
01340    5035 A7 84                 STA     ,X      set request no.
01350    5037 FC 508D               LDD     CHANEL  set channel no. & range
01360    503A ED 02                 STD     2,X
01370    503C AD 9F FBFA            JSR     [BIOS]  bios call
01371                       *
01375                       * output A/D converted data
01376                       *
01380    5040 E6 04                 LDB     4,X     get A/D converted data
01390    5042 4F                    CLRA
01400    5043 34 06                 PSHS    D       save data
01410    5045 30 8C 27              LEAX    <MES3-1,PCR message output
01420    5048 BD 9BDB               JSR     MESSAG
01430    504B 35 06                 PULS    D       data come back
01440    504D BD B615               JSR     D1OOUT  data output
01441                       *
01445                       * wait next data get timing
01446                       *
01450    5050 BD DB54               JSR     KEYIN   wait space key
01460    5053 81 20                 CMPA    #'      space ?
01470    5055 27 D9    5030         BEQ     DATAIN  yes,get next data
01480    5057 39                    RTS             no,end of getting data
01485                       *
01490                       * messages
01495                       *
01500    5058 0D            MES1    FCB     $0D,$0A
```

```
01510  505A     43              FCC   'Channel ='
01520  5063     00              FCC   0
01530  5064     20        MES2  FCC   '     Range ='
01540  506F     00              FCB   0
01550  5070     0D        MES3  FCB   $0D,$0A
01560  5072     20              FCC   ' A/D converted data ='
01570  5087     00              FCB   0
01575                     *
01580                     * working area
01585                     *
01590  5088     0005      RCB    RMB  5         only needs 5 bytes for rcb
01600  508D     00        CHANEL FCB  0         channel no.
01610  508E     48        RANGE  FCC  'H'       voltage range
01615                     *
01620           5000             END  START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=508E
PROGRAM ENTRY ADDR=5000


exec &H5000

Channel =0     Range =H
 A/D converted data =255
 A/D converted data =255
 A/D converted data =255
Ready
```

BIOS：MOTOR　－カセットモーターコントロール－

機　能　オーディオカセットのモーターをコントロールします．

パラメータ　Xレジスタ←RCB先頭アドレス

RCB＋0	リクエスト番号	＄０１
1		
2	モーターフラグ	＄ＦＦ…………モーターＯＮ ＄ＦＦ以外……モーターＯＦＦ

このリクエストはＲＣＢ領域としては３バイトしか必要としません．

解　説　モーターフラグの値によってリモート端子をコントロールします．このリクエストではエラーは生じません．

サンプル　０か１をキーインすることで，リモート端子をコントロールします．

0,1以外のkeyinはEND.

```
PAGE  001 (         ,         )

01000                              *
01010                              * MOTOR CONTROL
01020                              *                  (BIOS:MOTOR )
01030                                     OPT    M,NOS,P=255,NOO,NOG
01040                 9BDB         MESSAG EQU    $9BDB   output from (X+1) till 0
01050                 DB54         KEYIN  EQU    $DB54   key input with wait
01060                 D08E         OUT    EQU    $D08E   one character out
01070                 FBFA         BIOS   EQU    $FBFA   BIOS (extend-indirect)
01080                              *
01090  5000                               ORG    $5000
01100                 5000         START  EQU    *
01110                              *
01120                              * get motor flag
01130                              *
01140  5000 30        8C 22               LEAX   <PROMPT-1,PCR output message
01150  5003 BD        9BDB                JSR    MESSAG
01160  5006 BD        DB54                JSR    KEYIN   keyinput
01170  5009 81        30                  CMPA   #'0     keyinput must be '0'or'1'
01180  500B 27        04    5011          BEQ    M01
01190  500D 81        31                  CMPA   #'1
01200  500F 26        14    5025          BNE    M02     else,end
01210  5011 BD        D08E         M01    JSR    OUT     echoback
01220                              *
01230                              * motor control
01240                              *
01250  5014 80        32                  SUBA   #'1+1   1->$FF,0->$FE
01260  5016 30        8C 2E               LEAX   <RCB,PCR load rcb address
01270  5019 C6        01                  LDB    #$01    bios request MOTOR
01280  501B E7        84                  STB    ,X      set request no.
01290  501D A7        02                  STA    2,X     set motor flag
01300  501F AD        9F FBFA             JSR    [BIOS]  call bios
01310  5023 20        DB    5000          BRA    START
01320  5025 39                     M02    RTS
01330                              *
01340                              * messages
01350                              *
01360  5026           0D           PROMPT FCB    $0D,$0A
01370  5028           4D                  FCC    'Motor on =1 or Motor off =0 ?
01380  5046           00                  FCB    0
01390                              *
01400                              * working area
01410                              *
01420  5047           0003         RCB    RMB    3       only needs 3 bytes for rcb
```

```
01430                        *
01440            5000               END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5049
PROGRAM ENTRY ADDR=5000


exec &H5000

Motor on =1 or Motor off =0 ? 1
Motor on =1 or Motor off =0 ? 0
Motor on =1 or Motor off =0 ? 1
Motor on =1 or Motor off =0 ? 0
Motor on =1 or Motor off =0 ?
Ready
```

BIOS：CTBWRT　—カセットテープ１バイトライト—

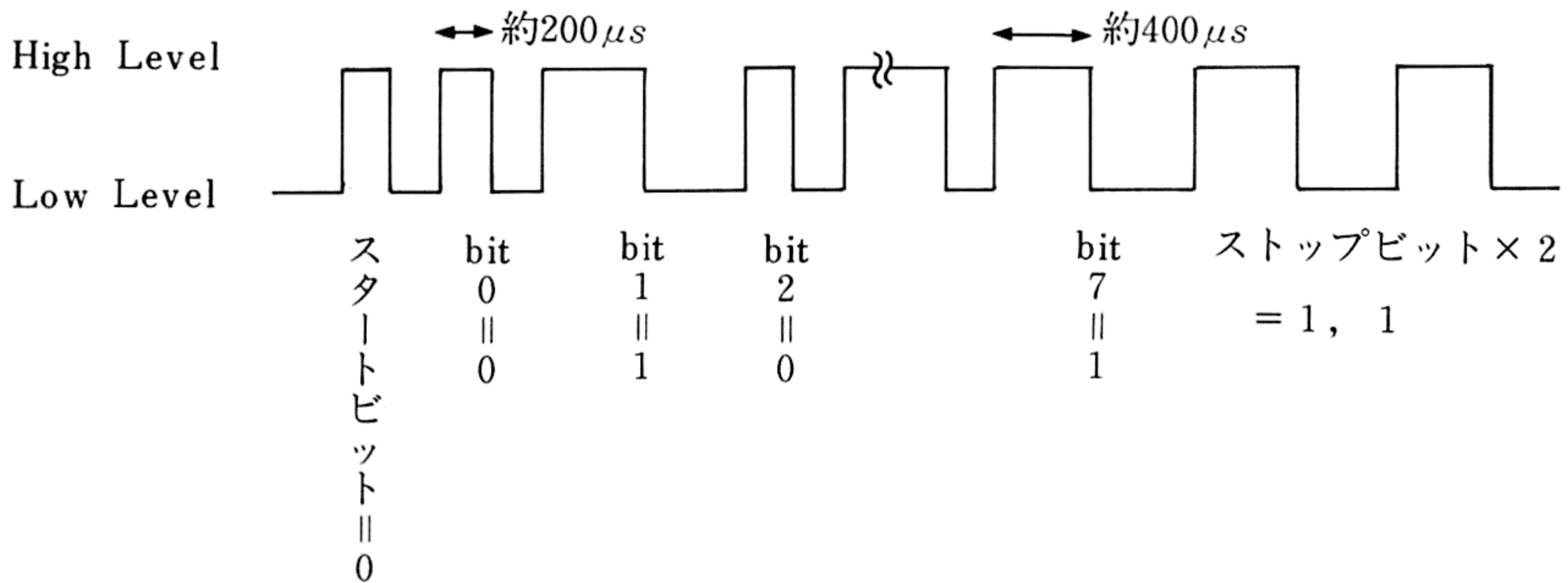
機　能　カセットテープに１バイトのデータを書き込む

パラメータ　　Ｘレジスタ←ＲＣＢ先頭アドレス

ＲＣＢ＋０	リクエスト番号	＄０２
１		
２	書き込みデータ	

このリクエストはＲＣＢ領域としては，３バイトしか必要ありません．

解　説　カセットテープに１バイトのデータを書き込みます．ルーチン内部で１．２ MHz か２ MHz かを検出して，いつも同じ波形になるようにしています．このリクエストではエラーは生じません．

＄ＡＡを出力した時の波形を示します．

BIOS：CTBRED －カセットテープ1バイトリードー

機　能　カセットテープから1バイト読み込みます．

パラメータ　Xレジスタ←RCB先頭アドレス

RCB＋0	リクエスト番号	＄03
1		
2		

このリクエストはRCB領域としては，3バイトしか必要ありません．

復帰情報　（RCB＋2）：読みこんだデータ

解　説　カセットテープから1バイト（8ビット）のデータを読み込みます．CTBWRT同様，ルーチンの内部でCPUのクロックを判別しています．

このリクエストは1バイトを正常に読み込むまで，Breakキーを押さないかぎりループしつづけます．（エラーも生じません．）

Breakキーが押された時には，＄D678（ABORTT）にジャンプして，F－BASICのコマンドレベルになります．

また，このリクエストを使用する場合には，時間的な考慮も必要です．

補足：FM7のテープ入出力全般にいえることですが，カセットテープレコーダーとの相性が悪いとなかなかデータを読み込まなくなります．この原因の一つに，テープレコーダーの同相・逆相というものがあります．FM7のテープリードルーチンは，このことを考慮していないので，読み込まない場合（逆相のとき）は，ケーブルの線を逆にすると良い場合があります．

サンプル　テープ中のデータを1バイトずつキャラクタで出力します．

```
PAGE  001 (         ,        )

01000                           *
01010                           * tape data output in character
01020                           *                       (BIOS:CTBRED)
01030                           *
01040                           * !!! not position independent !!!
01050                           *
01060                           * !!! must be CONSOLE 0,25,0,1 !!!
01070                           *
01080                                   OPT     M,NOO,NOS,P=255,NOG
01090                 9BDB      MESSAG  EQU     $9BDB    output from (X+1) till 0
01100                 FBFA      BIOS    EQU     $FBFA    BIOS (extend-indirect)
01110                           *
01120   5000                            ORG     $5000
01130                 5000      START   EQU     *
01140                           *
01150                           * output message
01160                           *
01170   5000 8E       502B              LDX     #MES-1   output message
01180   5003 BD       9BDB              JSR     MESSAG
01190                           *
```

```
01200                           * motor on
01210                           *
01220  5006 8E   503B                  LDX    #RCB0    load rcb for MOTOR ON
01230  5009 AD   9F FBFA               JSR    [BIOS]
01240                           *
01250                           * get tape data (1byte)
01260                           *
01270  500D 8E   503E           GET    LDX    #RCB1    load rcb for CTBRED
01280  5010 AD   9F FBFA               JSR    [BIOS]
01290                           *
01300                           * output data in hex
01310                           *
01320  5014 A6   02                    LDA    2,X
01330  5016 81   20                    CMPA   #'       spase !
01340  5018 25   04    501E            BCS    DOT      if asc<$1F or $7F
01350  501A 81   7F                    CMPA   #$7F     then output '.'
01360  501C 26   02    5020            BNE    ASC
01370  501E 86   2E             DOT    LDA    #'.
01380  5020 B7   5047           ASC    STA    CHR
01390  5023 8E   5041                  LDX    #RCB2    load rcb for OUTPUT
01400  5026 AD   9F FBFA               JSR    [BIOS]
01410  502A 20   E1    500D            BRA    GET
01420                           *
01430                           * message
01440                           *
01450  502C      0D0A           MES    FDB    $0D0A
01460  502E      74                    FCC    'tape data = '
01470  503A      00                    FCB    0
01480                           *
01490                           * rcb
01500                           *
01510  503B      01             RCB0   FCB    $01,0,$FF
01520  503E      03             RCB1   FCB    $03,0,0
01530  5041      14             RCB2   FCB    $14,0
01540  5043      5047                  FDB    CHR,1
01550  5047      00             CHR    FCB    0
01560                           *
01570            5000                  END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5047
PROGRAM ENTRY ADDR=5000


list

10 ' TEST PROGRAM FOR CTBRED
20 PRINT "TEST PROGRAM !!!"
30 GOTO 20

Ready
save"CAS0:TEST"

Ready
console 0,25,0,1


Ready
exec &H5000

tape data =

                          .<..TEST     ............ｱ

                                                                   .<.I    .I..:
■ TEST PROGRAM FOR CTBRED..ｒ..ｸ "TEST PROGRAM !!!"..ｮ..■ｼ ※ｷ.....J
.< .

Abort
Ready
```

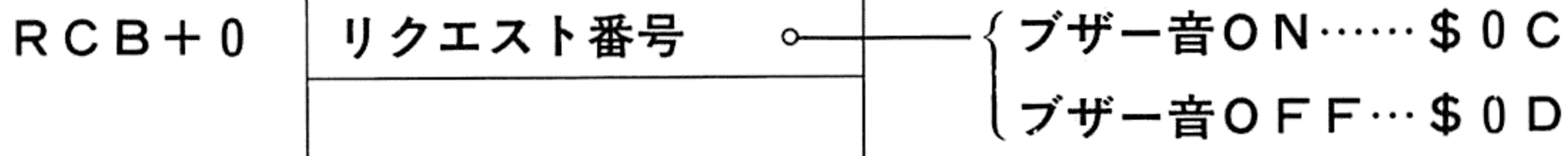
P.37

を PC 相対アドレッシングにして、RCB2+2の内容を書き換えるルーチンを付加すれば、ポジションインディペンダントになる。
後のKANJIRは LEAD ,PCR を使ってある。

BIOS：BEEPON・BEEPOF　ービープ音発生・ストップー

機　能　ブザー音をコントロールします．

パラメータ　　Ｘレジスタ←ＲＣＢ先頭アドレス

ＲＣＢ＋０	リクエスト番号	ブザー音ＯＮ……＄０Ｃ ブザー音ＯＦＦ…＄０Ｄ

このリクエストでは，ＲＣＢ領域としては２バイトしか必要としません．

解　説　Ｆ－ＢＡＳＩＣのＢＥＥＰ１，ＢＥＥＰ０に相当する動作をします．

サンプル　キー入力に応じてブザーをＯＮ・ＯＦＦします．

```
PAGE  001 (         ,          )

01000                          *
01010                          * BEEP CONTROL
01020                          *                (BIOS:BEEPON,BEEPOF)
01030                                 OPT    M,NOS,P=255,NOO,NOG
01040             9BDB         MESSAG EQU    $9BDB   output from (X+1) till 0
01050             DB54         KEYIN  EQU    $DB54   key input with wait
01060             D08E         OUT    EQU    $D08E   one character out
01070             FBFA         BIOS   EQU    $FBFA   BIOS (extend-indirect)
01080                          *
01090  5000                           ORG    $5000
01100             5000         START  EQU    *
01110                          *
01120                          * get request no.
01130                          *
01140  5000 30    8C 1F               LEAX   <PROMPT-1,PCR output message
01150  5003 BD    9BDB                JSR    MESSAG
01160  5006 BD    DB54                JSR    KEYIN   keyinput
01170  5009 BD    D08E                JSR    OUT     echoback
01180  500C C6    0C                  LDB    #$0C    bios request BEEPON
01190  500E 81    31                  CMPA   #'1     BEEPON='1' ?
01200  5010 27    06    5018          BEQ    BEEP    yes,RQNO.=$0C
01210  5012 5C                        INCB           bios request BEEPOF
01220  5013 81    30                  CMPA   #'0     BEEPOFF='0' ?
01230  5015 27    01    5018          BEQ    BEEP    yes,RQNO.=$0C
01240  5017 39                        RTS            no,end of execution
01250                          *
01260                          * beep control
01270                          *
01280  5018 30    8C 27        BEEP   LEAX   <RCB,PCR set rcb address
01290  501B E7    84                  STB    ,X      set request no.
01300  501D AD    9F FBFA             JSR    [BIOS]  call bios
01310  5021 20    DD    5000          BRA    START
01320                          *
01330                          * message
01340                          *
01350  5023       0D           PROMPT FCB    $0D,$0A
01360  5025       42                  FCC    'Beep on =1 or Beep off =0 ?
01370  5041       00                  FCB    0
01380                          *
01390                          * working area
01400                          *
01410  5042       0002         RCB    RMB    2       only needs 2 bytes for rcb
01420                          *
01430             5000                END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5043
```

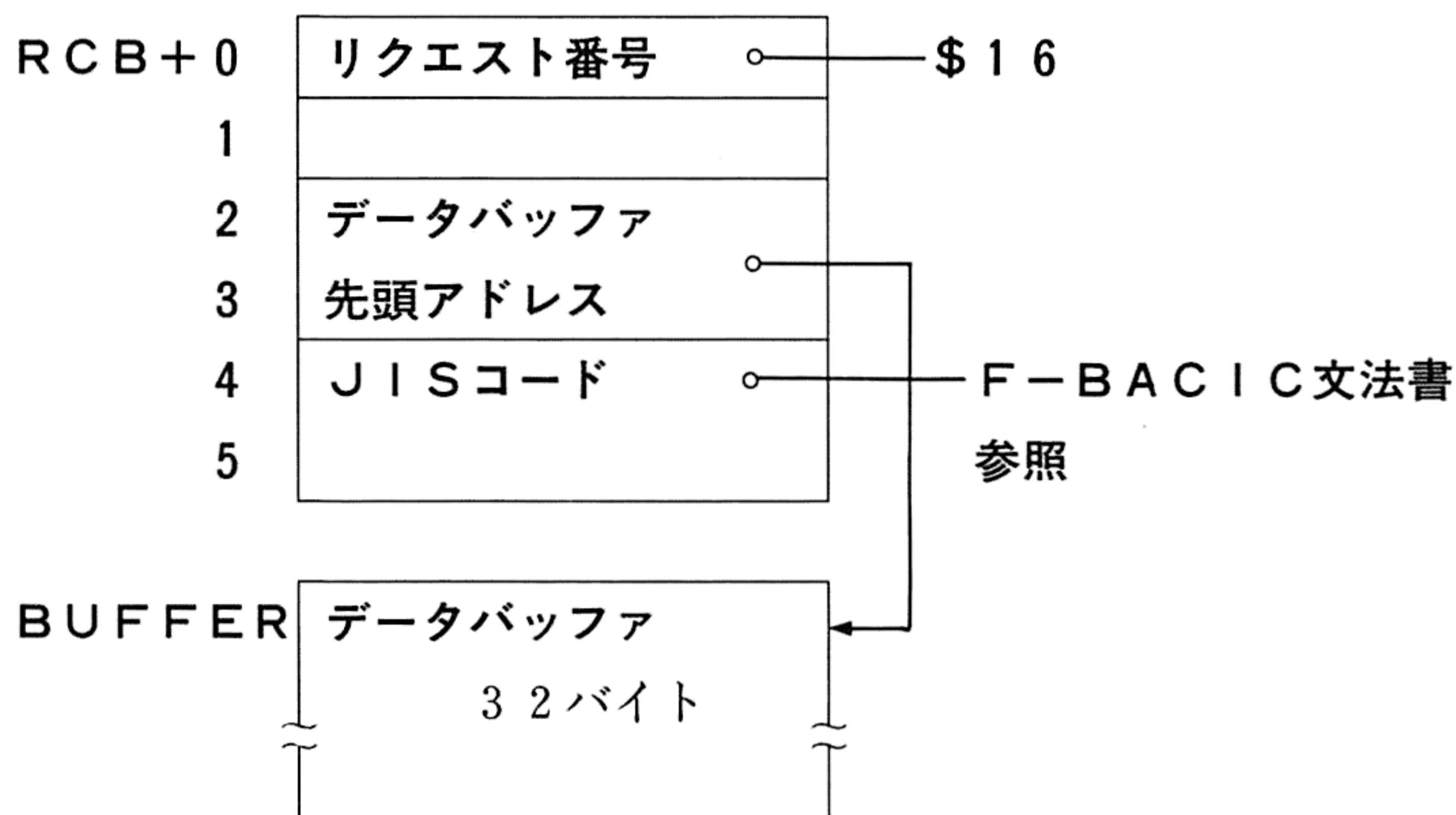
```
PROGRAM ENTRY ADDR=5000

exec &H5000

Beep on =1 or Beep off =0 ? 1
Beep on =1 or Beep off =0 ? 0
Beep on =1 or Beep off =0 ? 1
Beep on =1 or Beep off =0 ? 0
Beep on =1 or Beep off =0 ?
Ready
```

BIOS：KANJIR　－漢字ＲＯＭデータリード－

機　能　漢字ＲＯＭから指定されたＪＩＳコードの漢字のドットパターンを取り出す．

パラメータ　　Ｘレジスタ←ＲＣＢ先頭アドレス

ＲＣＢ＋０	リクエスト番号	＄１６
１		
２	データバッファ	
３	先頭アドレス	→ BUFFER
４	ＪＩＳコード	Ｆ－ＢＡＣＩＣ文法書参照
５		

ＢＵＦＦＥＲ	データバッファ ３２バイト

このリクエストは，ＲＣＢ領域として６バイトしか必要ありませんが，ドットパターンの格納用に３２バイトのバッファをＲＣＢと別に確保の必要があります．

復帰情報　漢字ＲＯＭから指定されたＪＩＳコードの漢字のドットパターンを３２バイトのバッファにセットします．

解　　説　ＪＩＳコードについては「Ｆ－ＢＡＳＩＣ文法書」，ＲＯＭのアドレスは，「ユーザーズマニュアルシステム仕様」を御参照下さい．

ＪＩＳコードにないものを指定した場合には，空白（＄００）でデータバッファをクリアします．よってエラーは生じません．

サンプル　漢字ＲＯＭのフォントを表示します．

```
100 '
110 ' KANJI-ROM read & font display
120 '                  (BASIC)
130 INPUT "JIS CODE =$",CODE$
140 IF CODE$="" THEN END
150 CODE=VAL("&H"+CODE$)
160 IF CODE<0 THEN CODE=CODE+65536!
170 POKE &H5002,INT(CODE/256):            'set JIS code (high)
180 POKE &H5003,CODE-INT(CODE/256)*256:   '              (low)
190 EXEC &H5000
200 PRINT"----------------"
210 GOTO 130
```

```
PAGE  001 (        ,        )

01000                          *
01010                          * read KANJI ROM & font display
01020                          *             (BIOS:KANJIR)
01030                                 OPT   M,NOS,P=255,NOO,NOG
01040              9B50        CRLF   EQU   $9B50   output CR & LF
01050              D08E        OUT    EQU   $D08E   one character output
01060              FBFA        BIOS   EQU   $FBFA   BIOS (extend-indirect)
01070                          *
01080  5000                           ORG   $5000
01090              5000        START  EQU   *
01100  5000 20     02    5004         BRA   KANJI
01110                          *
01120                          * JIS code (set by BASIC)
01130                          *
01140  5002        0000        CODE   FDB   0
01150                          *
01160                          * get kanji font
01170                          *
01180  5004 30     8C 44       KANJI  LEAX  <RCB,PCR load rcb address
01190  5007 86     16                 LDA   #$16    bios request KANJIR
01200  5009 A7     84                 STA   ,X      set request no.
01210  500B 33     8C 43              LEAU  <DATA,PCR set data buffer
01220  500E EF     02                 STU   2,X
01230  5010 EC     8C EF              LDD   <CODE,PCR set JIS code
01240  5013 ED     04                 STD   4,X
01250  5015 AD     9F FBFA            JSR   [BIOS]  call bios
01260                          *
01270                          * font display
01280                          *
01290  5019 BD     9B50               JSR   CRLF
01300  501C CE     0010               LDU   #16     loop counter
01310  501F 30     8C 2F              LEAX  <DATA,PCR load font data address
01320  5022 E6     80          LOOP   LDB   ,X+     font out (16bit)
01330  5024 8D     10    5036         BSR   DOTOUT  left 8 bit
01340  5026 E6     80                 LDB   ,X+
01350  5028 8D     0C    5036         BSR   DOTOUT  right 8 bit
01360  502A BD     9B50               JSR   CRLF
01370  502D 33     5F                 LEAU  -1,U
01380  502F 1183   0000               CMPU  #0      16 line output ?
01390  5033 26     ED    5022         BNE   LOOP
01400  5035 39                        RTS
01410                          *
01420                          * subroutine: dot pattern output
01430                          *
01440  5036 34     04          DOTOUT PSHS  B       save dotpattern
01450  5038 C6     08                 LDB   #8      output 8 bit
01460  503A 86     2A          DO1    LDA   #'*
01470  503C 68     E4                 LSL   ,S      bit on ?
01480  503E 25     02    5042         BCS   DO2     yes,print '*'
01490  5040 86     A5                 LDA   #'.     no,print '.'
01500  5042 BD     D08E        DO2    JSR   OUT
01510  5045 5A                        DECB
01520  5046 26     F2    503A         BNE   DO1
01530  5048 35     04                 PULS  B       drop stack top
01540  504A 39                        RTS
01550                          *
01560                          * working area
01570                          *
01580  504B        0006        RCB    RMB   6       only needs 6 bytes for rcb
01590  5051        0020        DATA   RMB   32      data buffer
01600                          *
01610              5000               END   START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5070
PROGRAM ENTRY ADDR=5000


RUN
JIS CODE =$4959
```

← BASICプログラムのPOKE命令でセットされている。

最初にAに＊を入れておいて必要なら・に変更

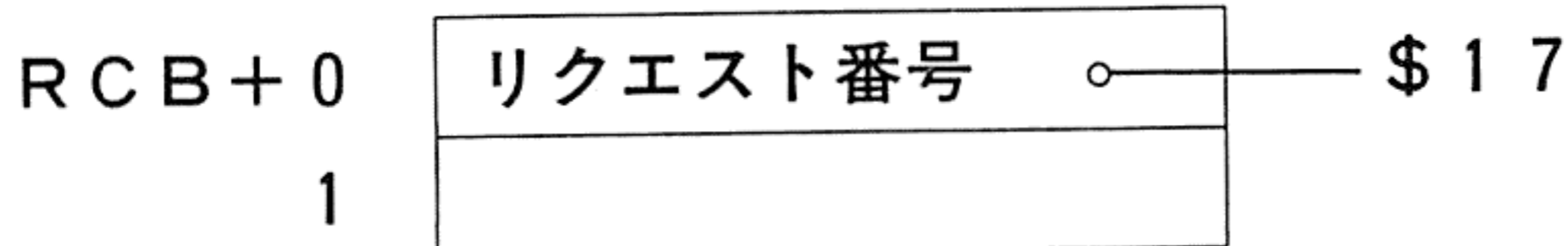
(判定後＊か・のいずれかを代入するより能率が悪い)

```
.......*........
***************.
*.............*.
*.***********.*.
................
...*********....
...*.......*....
...*********....
................
..***********...
..*....*....*...
..*....*....*...
..***********...
..*....*....*...
..*....*....*...
..***********...
----------------
JIS CODE =$

Ready
HARDC
```

BIOS：LPCHK　－プリンタ・レディチェック－

機　能　プリンタが正常であるかどうかをチェックします．

パラメータ　Ｘレジスタ←ＲＣＢ先頭アドレス

ＲＣＢ＋０	リクエスト番号	＄１７
１		

このリクエストは，ＲＣＢ領域として２バイトしか必要としません．

復帰情報　（ＲＣＢ＋１）：プリンタのステータス（エラー番号）

＄３２（５０）：プリンタが用紙切れをおこしている．
＄３３（５１）：プリンタがビジー状態であるか，エラーをおこしている．
＄００（　０）：プリンタは正常である．

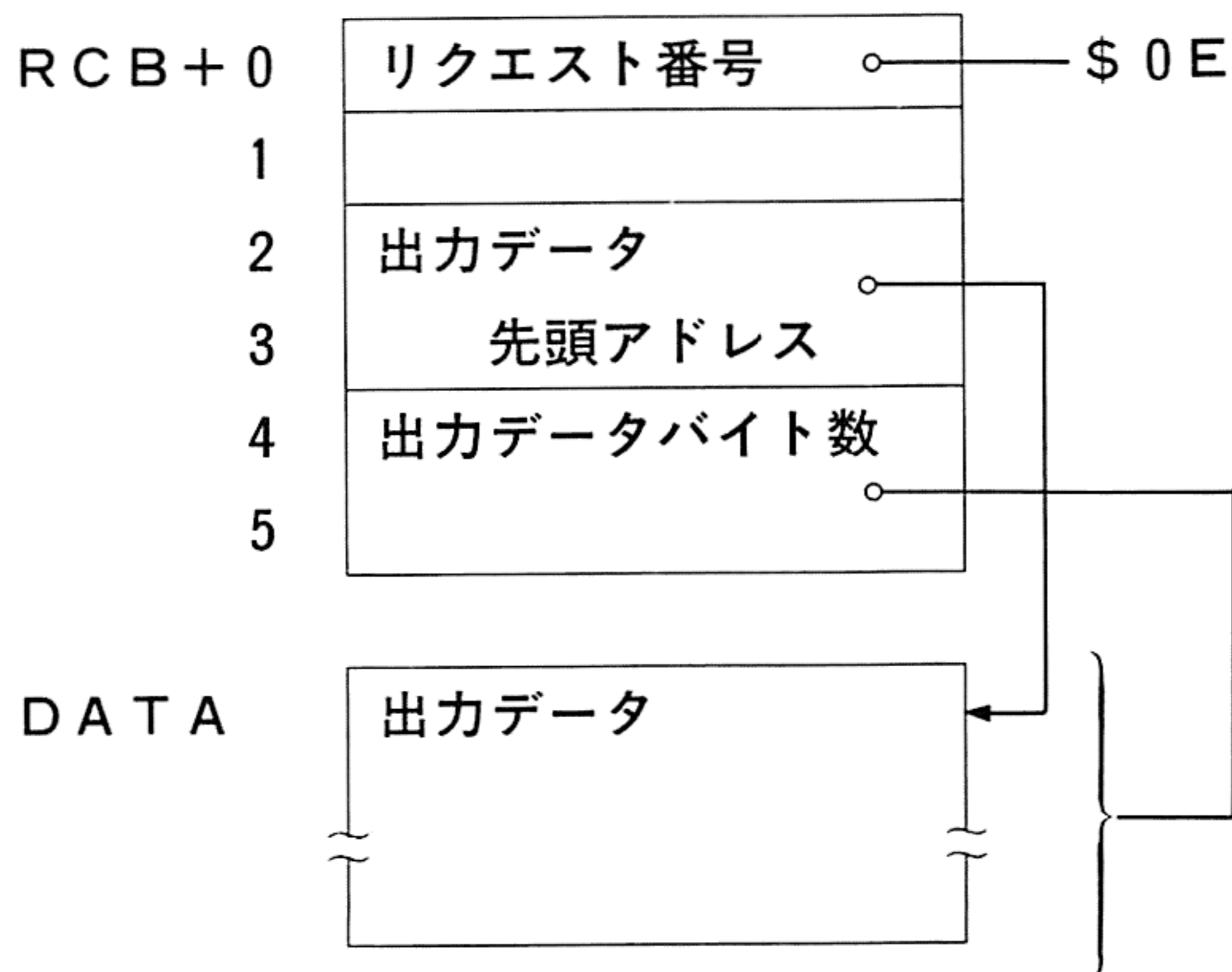
解　説　このリクエストでは，まずエラーをチェックし，エラーでない時は，プリンタがレディになるまで約２秒待ちます．約２秒経過しても，ビジー状態が続く場合には，エラーとします．

他のプリンタ関係のリクエストは，プリンタのチェックを行わないので，事前にこのリクエストで，プリンタの状態を知る必要があります．

BIOS：LPOUT　－プリンタ出力－

機　能　プリンタに対して，文字列を出力します．

パラメータ　Ｘレジスタ←ＲＣＢ先頭アドレス

ＲＣＢ＋０	リクエスト番号	＄０Ｅ
１		
２	出力データ	→ DATA
３	先頭アドレス	
４	出力データバイト数	→ DATA の長さ
５		
ＤＡＴＡ	出力データ	

このリクエストは，ＲＣＢ領域としては６バイトしか必要としません．

解　説　プリンタに対して文字列を出力します．このリクエストは単に1バイトずつプリンタに送出するだけなので，改行などのコードは出力データ中に用意する必要があります．

また，出力中に用紙切れなどのエラーが生じても，このリクエストは処置を行わず，レディになるのを待ちます．ただし，Breakキーが押された時は，＄D678にジャンプしてＢＡＳＩＣのコマンドレベルへ戻ります．

サンプル　プリンタをチェックして，文字列を出力します．

```
PAGE   001 (         ,         )

01000                          *
01010                          * PRINTER check & output message to PRINTER
01020                          *           (BIOS:LPCHK,LPOUT)
01030                                  OPT     M,NOS,P=255,NOO,NOG
01040               9BDB       MESSAG EQU     $9BDB    output from (X+1) till 0
01050               FBFA       BIOS   EQU     $FBFA    BIOS (extend-indirect)
01060                          *
01070  5000                            ORG     $5000
01080               5000       START  EQU     *
01090                          *
01100                          * printer check
01110                          *
01120  5000 30     8C 25               LEAX    <LPCHK,PCR load rcb address
01130  5003 AD     9F FBFA             JSR     [BIOS]   printer check
01140  5007 E6     01                  LDB     1,X      error detected ?
01150  5009 27     15    5020          BEQ     READY
01160  500B 30     8C 44               LEAX    <PAPER-1,PCR
01170  500E C1     32                  CMPB    #50      paper empty ?
01180  5010 27     0A    501C          BEQ     ERR
01190  5012 30     8C 55               LEAX    <NREADY-1,PCR
01200  5015 C1     33                  CMPB    #51      not ready ?
01210  5017 27     03    501C          BEQ     ERR
01220                          *
01230                          * output message to printer
01240                          *
01250  5019 30     8C 66               LEAX    <BIOSER-1,PCR not printer error
01260  501C BD     9BDB        ERR     JSR     MESSAG   error message output
01270  501F 39                         RTS
01280  5020 30     8C 07       READY   LEAX    <LPOUT,PCR get rcb address
01290  5023 AD     9F FBFA             JSR     [BIOS]   printer output
01300  5027 39                         RTS
01310                          *
01320                          * rcb for LPCHK
01330                          *       only needs 2 bytes for rcb
01340  5028         1700       LPCHK   FDB     $1700    bios request LPCHK (No.=$17)
01350                          *
01360                          * rcb for LPOUT
01370                          *       only needs 6 bytes for rcb
01380  502A         0E00       LPOUT   FDB     $0E00    bios request LPOUT (No.=$0E)
01390  502C         5030               FDB     DATA     output data address
01400  502E         0023               FDB     DATAE-DATA output data length
01410  5030         0D0A       DATA    FDB     $0D0A    output data
01420  5032         6F                 FCC     'out put message for printer !!!'
01430  5051         0D0A               FDB     $0D0A
01440               5053       DATAE   EQU     *
01450                          *
01460                          * error messages
01470                          *
01480  5053         0D0A       PAPER   FDB     $0D0A    error messages
01490  5055         70                 FCC     'paper empty error !!!'
01500  506A         00                 FCB     0
01510  506B         0D0A       NREADY  FDB     $0D0A
01520  506D         70                 FCC     'printer not ready !!!'
01530  5082         00                 FCB     0
01540  5083         0D0A       BIOSER  FDB     $0D0A
```

```
01550  5085       42                FCC     'BIOS error !!!'
01560  5093       00                FCB     0
01570                      *
01580             5000              END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5093
PROGRAM ENTRY ADDR=5000
```

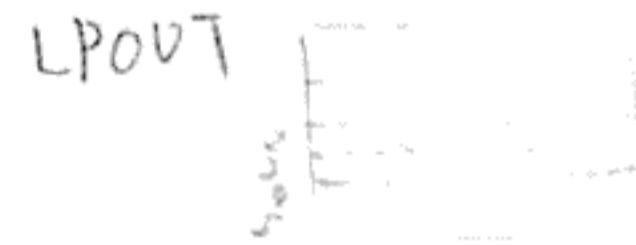
プリンタ出力

```
out put message for printer !!!
```

ＣＲＴ出力

```
exec &H5000

Ready
exec &H5000

printer not ready !!!
Ready
```

何故かは解らないが、$32, $33 以外のエラーもあるらしい。
BIOS ERROR !!! を出力する。

FM-7 TYPEWRITER

LPOUT

```
        LEAX   LPOUT, PCR
        LDU    2,X
        JSR    KEYIN
        LDB    5,X
        INCB
        BMI    BUFERROR
        STB    5,X
        STA    B,U
START   LEAX   LPCHK,PCR
        JSR    [BIOS]
        LDB    1,X         CMP B   #$33
        :                  BEQ     NEXT
REDY    LEAX   LPOUT,PCR
        JSR    [BIOS]
        CLR    5,X
        BRA    NEXT
BUFERROR LEAX  BUFER-1,PCR
        JSR    MESSAGE
        RTS
BUFER   FDB    $0D0A
        FCC    /BUFFER_ERROR_!/
        FCB    $00
```

BIOS：HDCOPY　－ＣＲＴスクリーン・イメージコピー(1)－

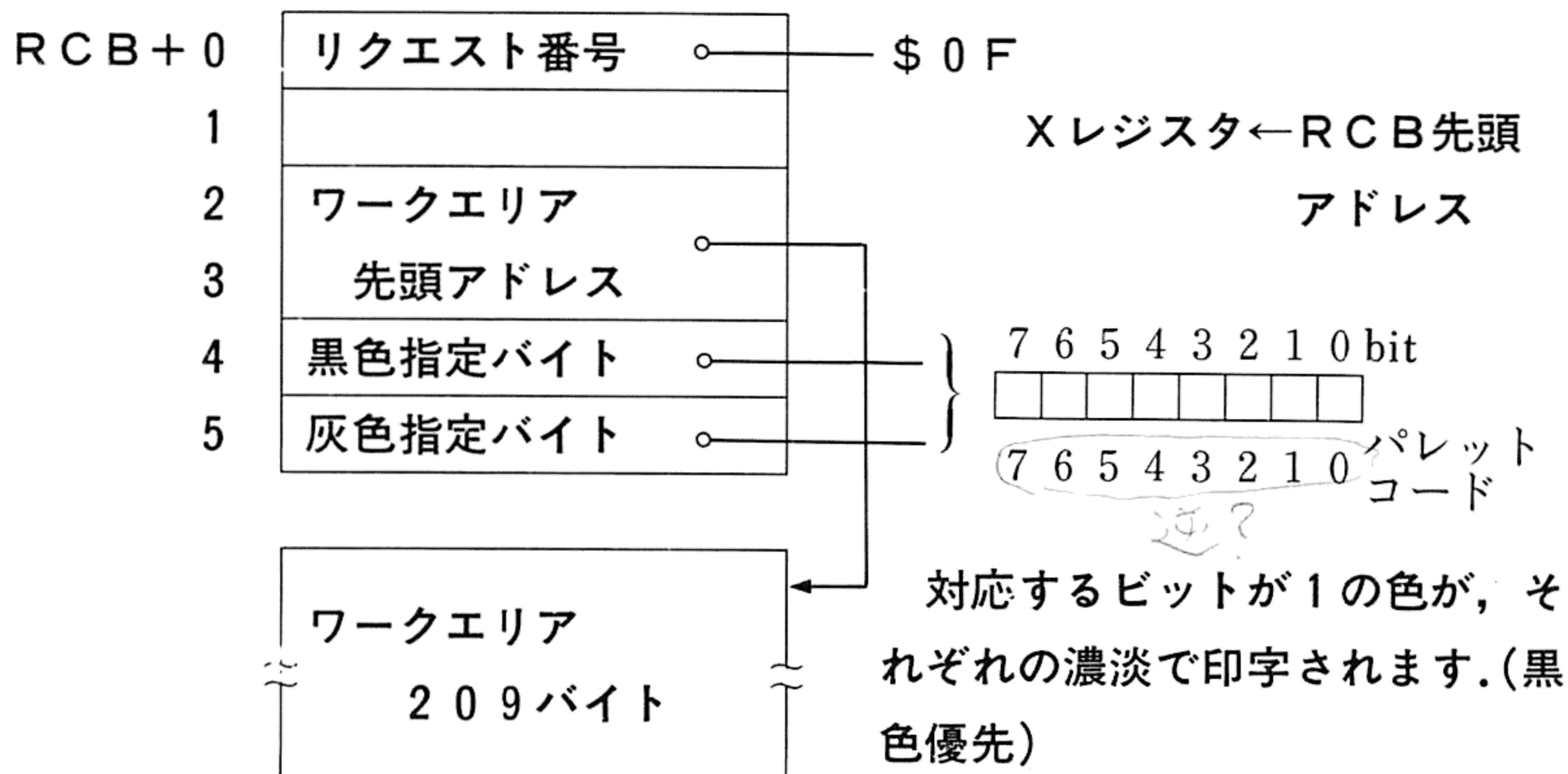
機　　能　ＣＲＴ画面をプリンタにコピーします．

パラメータ　このリクエストは，ＲＣＢ領域（６バイトのみ必要）以外に，２０９（＄Ｄ１）バイトのワークエリアを必要とします．

RCB	内容	
ＲＣＢ＋０	リクエスト番号	＄０Ｆ
１		
２	ワークエリア	
３	先頭アドレス	→ ワークエリア　２０９バイト
４	黒色指定バイト	
５	灰色指定バイト	

Ｘレジスタ←ＲＣＢ先頭アドレス

7	6	5	4	3	2	1	0 bit
7	6	5	4	3	2	1	0 パレットコード

対応するビットが１の色が，それぞれの濃淡で印字されます．(黒色優先)

解　　説　ＣＲＴ画面をプリンタにコピーします．このコピーでは画面上の１ドットをプリンタの４ドットに拡大して印字するため，３レベル（黒灰白）の濃淡をつけることができます．Ｆ－ＢＡＳＩＣのＨＡＲＤＣ１が利用しています．

このリクエストは，プリンタの状態チェック等は行わず，プリンタにエラーが生じると，回復するまでループしつづけます．Breakキーが押された時には，ＢＡＳＩＣのコマンドレベルに制御が移ります．またこのリクエストは一部でＦＩＲＱをマスクしています．

BIOS：SCREEN　－ＣＲＴスクリーン・イメージコピー(2)－

機　　能　ＣＲＴ画面をプリンタにコピーします．

パラメータ　灰色指定バイトがない以外はＨＤＣＯＰＹと同様です．

解　　説　このコピーは画面上の１ドットとプリンタの１ドットを対応させて印字しますので，２レベル（黒白）の濃淡がつけられます．ＦＢＡＳＩＣのＨＡＲＤＣ２が利用しています．その他は，ＨＤＣＯＰＹを御参照下さい．

補足：このリクエストに限って，ＦＭ８との互換性はありません．ＦＭ８のソフトを利用する場合などには注意して下さい（ＦＭ８では濃淡の指定はできません）．

サンプル　ＨＤＣＯＰＹを使用したサンプルです．

```
PAGE  001 (        ,        )

01000                         *
01010                         * SCREEN HARD COPY
01020                         *                 (BIOS:HDCOPY)
01030                                OPT    M,NOS,NOG,NOO,PAGE=255
01040            FBFA         BIOS   EQU    $FBFA    BIOS (extend-indirect)
01050                         *
01060  5000                          ORG    $5000
01070            5000         START  EQU    *
01080                         *
01090  5000 20   02    5004          BRA    HDCOPY
01100                         *
01110                         * shades code
01120                         *
01130  5002      E0           BLACK  FCB    %11100000 output ****
01140  5003      1E           GRAY   FCB    %00011110 output .*.*
01150                         *
01160                         * set rcb
01170                         *
01180  5004 30   8C 13        HDCOPY LEAX   <RCB,PCR load rcb address
01190  5007 86   0F                  LDA    #$0F     bios request HDCOPY
01200  5009 A7   84                  STA    ,X       set request no.
01210  500B 33   8C 12               LEAU   <WORK,PCR set working area
01220  500E EF   02                  STU    2,X
01230  5010 EC   8C EF               LDD    <BLACK,PCR set shades code
01240  5013 ED   04                  STD    4,X
01241                         *
01242                         * screen hard copy
01243                         *
01250  5015 AD   9F FBFA             JSR    [BIOS]   call bios
01260  5019 39                       RTS
01270                         *
01280                         * working area
01290                         *
01300  501A      0006         RCB    RMB    6        only needs 6 bytes for rcb
01310  5020      00D1         WORK   RMB    209      working area for bios
01320                         *
01330            5000                END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50F0
PROGRAM ENTRY ADDR=5000
```

color = 0

color = 1

color = 2

color = 3

color = 4

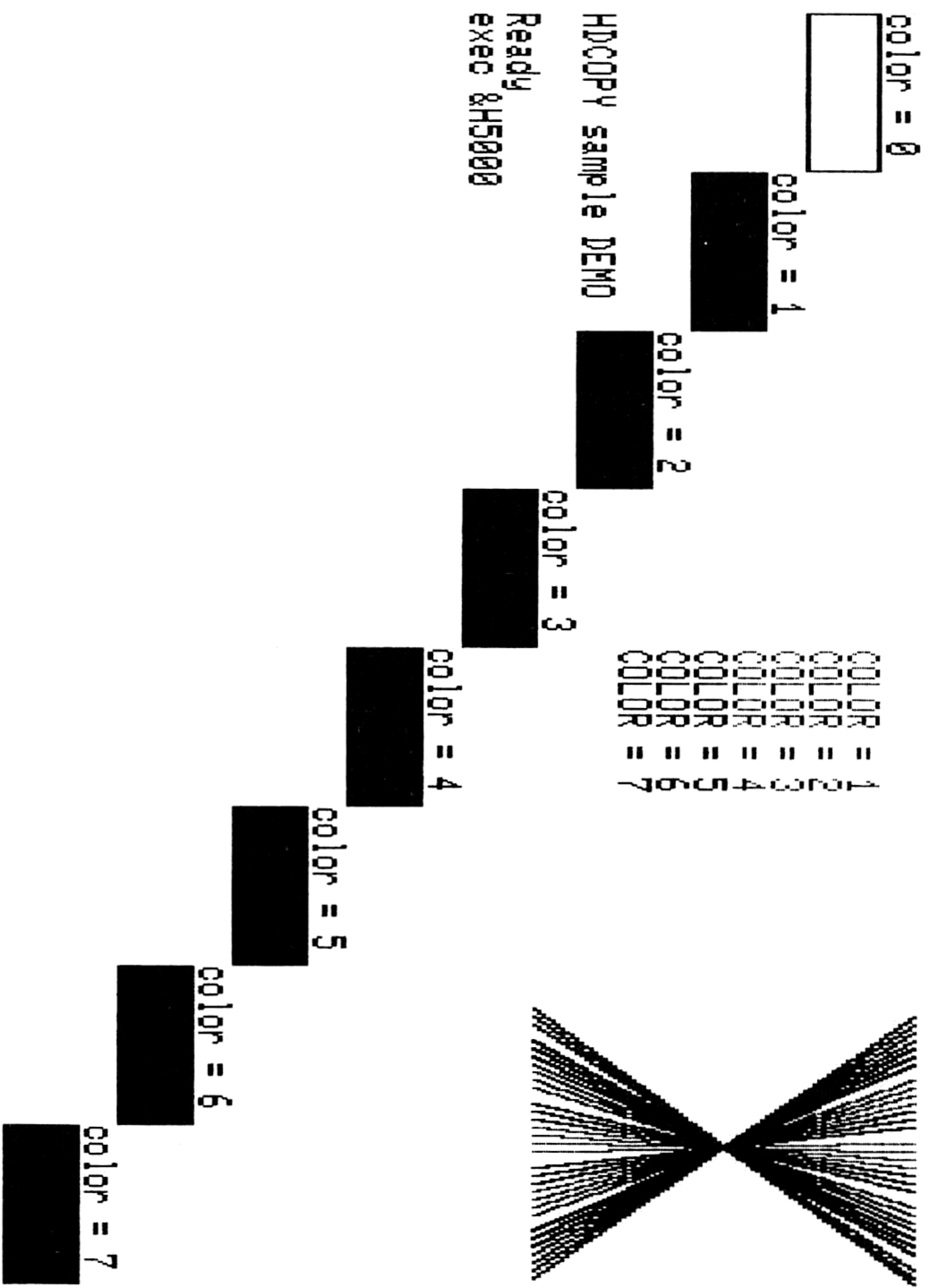
color = 5

color = 6

color = 7

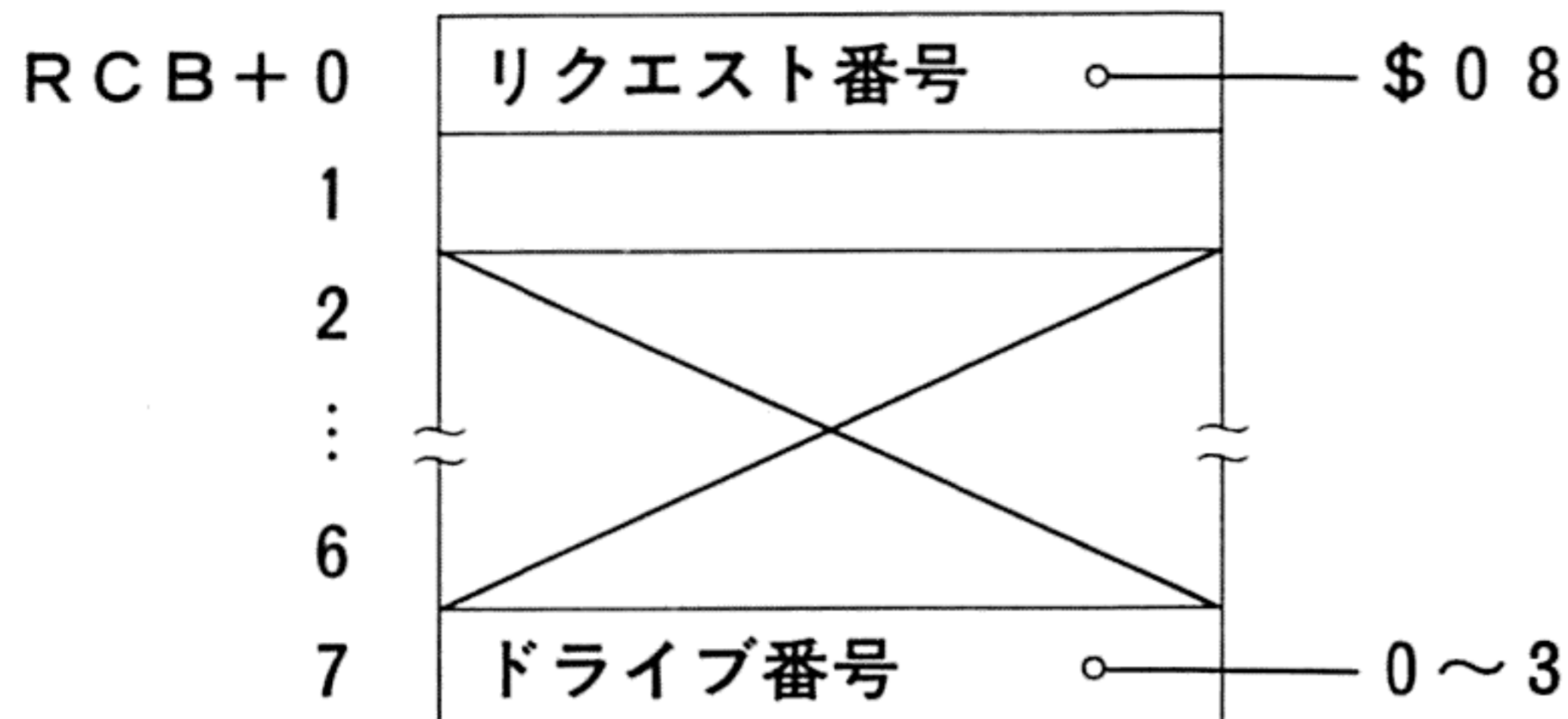
COLOR = 1
COLOR = 2
COLOR = 3
COLOR = 4
COLOR = 5
COLOR = 6
COLOR = 7

HDCOPY sample DEMO

Ready
exec &H5000

BIOS：RESTOR －シークトラック０－

機　能　ディスクに対して，シークトラック０を実行します．

パラメータ　Ｘレジスタ←ＲＣＢ先頭アドレス

ＲＣＢ＋０	リクエスト番号	＄０８
１		
２		
⋮		
６		
７	ドライブ番号	０～３

復帰情報　（ＲＣＢ＋１）：エラー番号

＄００（　０）	エラーなし
＄０Ａ（１０）	ディスク・ノット・レディ
＄０Ｂ（１１）	ディスクライトプロテクテッド
＄０Ｃ（１２）	ハードエラー
＄０Ｄ（１３）	ＣＲＣエラー
＄０Ｅ（１４）	Deleted Data Mark検出
＄０Ｆ（１５）	タイムオーバーエラー

このうち，このリクエストでは，＄０Ａ，＄０Ｆのエラーが生じます．それ外外は，ＤＷＲＩＴＥ，ＤＲＥＡＤによって生じるものです．

解　説　指定されたドライブのヘッドをトラック０へ移動します．（ＦＤＣに対してリストアコマンドを送出する．）このときディスクのモーターが停止していた場合には，モーターを動かして安定するまで約２秒間待ったのちにリストア動作を行います．

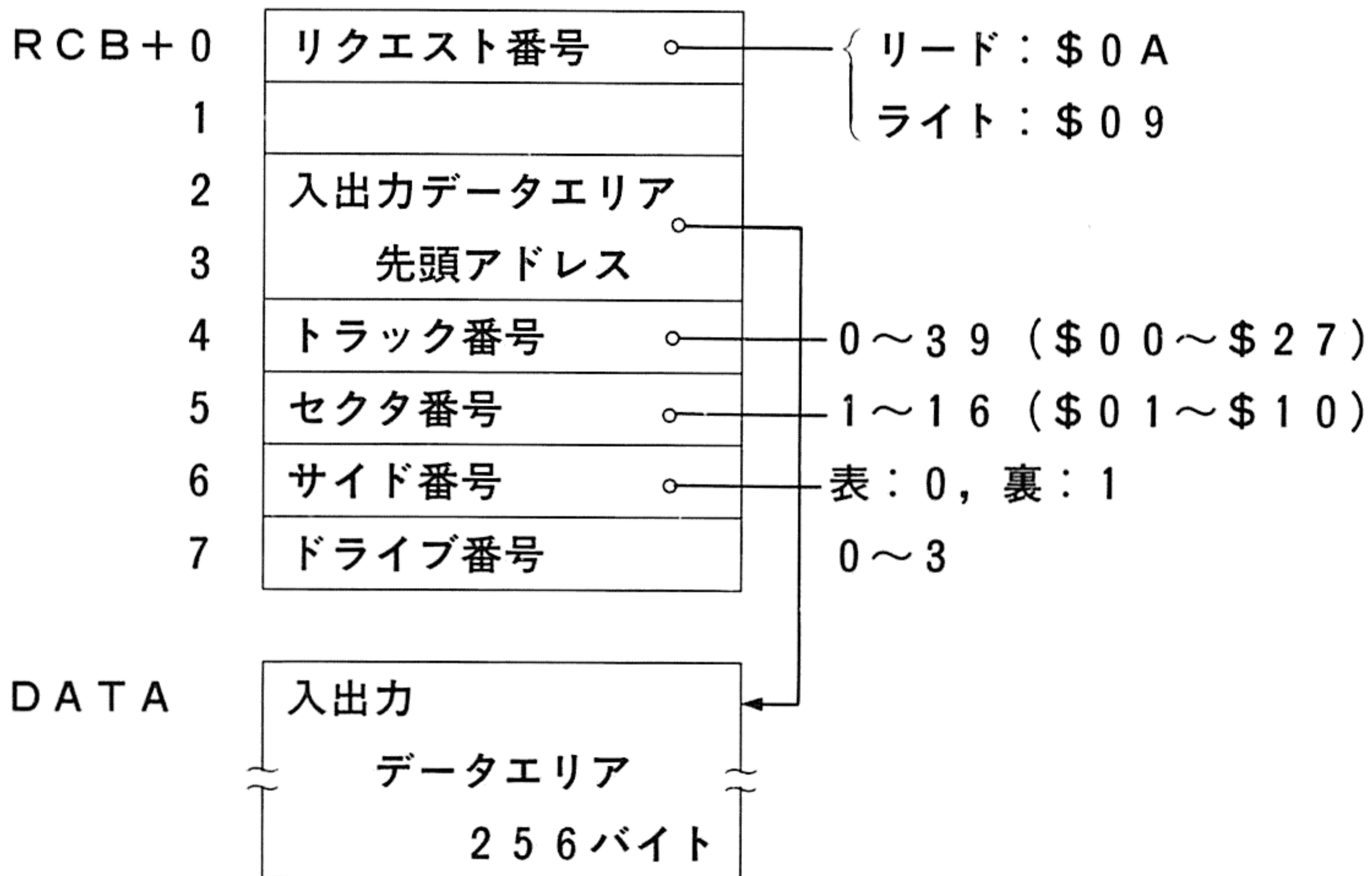
BIOS：DREAD・DWRITE　―ディスク1セクタリード・ライト―

機　　能　ディスクに対して１セクタ分のリード・ライトを行います．

パラメータ　このリクエストはＲＣＢ領域の他に入出力データエリアとして２５６バイトのメモリ領域を必要とします．

Ｘレジスタ←ＲＣＢ先頭アドレス

ＲＣＢ＋０	リクエスト番号	リード：＄０Ａ ライト：＄０９
１		
２	入出力データエリア	→ DATA
３	先頭アドレス	
４	トラック番号	０～３９（＄００～＄２７）
５	セクタ番号	１～１６（＄０１～＄１０）
６	サイド番号	表：０，裏：１
７	ドライブ番号	０～３

ＤＡＴＡ	入出力データエリア２５６バイト

復帰情報　ＲＣＢ＋１番地にエラー番号がセットされます．（エラー番号についてはＲＥＳＴＯＲの項を参照）

解　　説　ディスクに対して１セクタ分（＝２５６バイト）のリード・ライトを行います．このリクエストでエラーが生じた場合には，ユーザー側でリトライ（再実行）などの処置をしなければなりません．このとき再度ＲＥＳＴＯＲを呼び出した後に行うと良い場合があります．

このリクエストではリード・ライトの際，ＩＲＱ，ＦＩＲＱをマスクしているので注意して下さい（リード・ライト中はBreakキーは効きません）．

サンプル　指定したセクタを，＄６０００～＄６０ＦＦに取り出します．

```
PAGE  001 (          ,          )

01000                    *
01010                    * DISK READ & WRITE
01020                    *      (BIOS:RESTOR,DWRITE,DREAD)
01030                           OPT    M,NOS,NOG,NOO,PAGE=255
01040           9BDB     MESSAG EQU    $9BDB   output from (X+1) till 0
01050           D08E     OUT    EQU    $D08E   one character output
01060           FBFA     BIOS   EQU    $FBFA   bios (extend-indirect)
01070           AC3D     A16OUT EQU    $AC3D   output A req in HEX
```

```
01080              DB54       KEYIN  EQU    $DB54    keyinput with wait
01090                         *
01100   5000                         ORG    $5000
01110              5000       START  EQU    *
01120                         *
01130   5000 31    8D 00EE           LEAY   RCB,PCR load rcb address
01131                         *
01132                         * set drive no.
01133                         *
01140   5004 30    8D 0093    DRIVE  LEAX   MES1-1,PCR output message
01150   5008 BD    9BDB              JSR    MESSAG
01160   500B 8D    62    506F        BSR    GETNUM   get number
01170   500D C1    03                CMPB   #3       drive no. = 0..3
01180   500F 22    F3    5004        BHI    DRIVE
01190   5011 E7    27                STB    7,Y      set drive no.
01191                         *
01192                         * disk restore (seek track 0)
01193                         *
01200   5013 86    08                LDA    #$08     bios request RESTOR
01210   5015 A7    A4                STA    ,Y       set request no.
01220   5017 1F    21                TFR    Y,X      set rcb address
01230   5019 AD    9F FBFA           JSR    [BIOS]   call bios
01240   501D 25    70    508F        BCS    ERROR    if error detected
01241                         *
01242                         * set track no.
01243                         *
01250   501F 30    8D 0086    TRACK  LEAX   MES2-1,PCR output message
01260   5023 BD    9BDB              JSR    MESSAG
01270   5026 8D    47    506F        BSR    GETNUM   get number
01280   5028 C1    27                CMPB   #39      track no. = 0..39
01290   502A 22    F3    501F        BHI    TRACK
01300   502C E7    24                STB    4,Y      set track no.
01301                         *
01302                         * set sector no. & side no.
01303                         *
01310   502E 30    8D 0085    SECTOR LEAX   MES3-1,PCR output message
01320   5032 BD    9BDB              JSR    MESSAG
01330   5035 8D    38    506F        BSR    GETNUM   get track no. (0..32)
01340   5037 86    00                LDA    #0       assume side no.=0
01350   5039 5D                      TSTB            track no.=0 ?
01360   503A 27    E3    501F        BEQ    TRACK    redo from start
01370   503C C1    10                CMPB   #16      side 0 ?
01380   503E 23    07    5047        BLS    SIDE     yes,side 0 sector 1-16
01390   5040 C1    20                CMPB   #32      side 1 ?
01400   5042 22    DB    501F        BHI    TRACK    no,redo from start
01410   5044 C0    10                SUBB   #16      yes,side 1 sector 1-16
01420   5046 4C                      INCA            side no.=1
01430   5047 A7    26         SIDE   STA    6,Y      set side no.
01440   5049 E7    25                STB    5,Y      set sector no.
01441                         *
01442                         * set buffer addr. & request no.
01443                         *
01450   504B CC    6000              LDD    #BUFFER set buffer address
01460   504E ED    22                STD    2,Y
01470   5050 30    8D 0071           LEAX   MES4-1,PCR output message
01480   5054 BD    9BDB              JSR    MESSAG
01490   5057 BD    DB54              JSR    KEYIN    keyinput
01500   505A BD    D08E              JSR    OUT      echoback
01510   505D C6    0A                LDB    #$0A     assume bios request DREAD
01520   505F 81    77                CMPA   #'w      write ?
01530   5061 26    01    5064        BNE    RQNO     no,request no.=$0A(DREAD)
01540   5063 5A                      DECB            bios request DWRITE($09)
01550   5064 E7    A4         RQNO   STB    ,Y       set request no.
01560   5066 1F    21                TFR    Y,X      set rcb address
01570   5068 AD    9F FBFA           JSR    [BIOS]   call bios
01580   506C 25    21    508F        BCS    ERROR    if error detected
01590   506E 39                      RTS
01600                         *
01610                         * subroutine:get number (0..255) in Breg
01620                         *
01630   506F 5F               GETNUM CLRB
01640   5070 4F                      CLRA
01650   5071 34    02         GN01   PSHS   A        save keyin number
01660   5073 86    0A                LDA    #10      Breg=Breg*10
```

```
01670  5075 3D                     MUL
01680  5076 EB   E0                ADDB   ,S+      Breg=Breg+keyin_number
01690  5078 34   04                PSHS   B
01700  507A BD   DB54              JSR    KEYIN    keyin
01710  507D 35   04                PULS   B
01720  507F 81   30                CMPA   #'0      test '0'..'9'
01730  5081 25   0B    508E        BCS    GNO2
01740  5083 81   39                CMPA   #'9
01750  5085 22   07    508E        BHI    GNO2
01760  5087 BD   D08E              JSR    OUT      echo back
01761  508A 80   30                SUBA   #'0      character into number
01762  508C 20   E3    5071        BRA    GNO1
01763  508E 39              GNO2   RTS             Breg=number 0..255
01770                       *
01771                       * error message output
01772                       *
01780  508F 30   8D 004B    ERROR  LEAX   ERRMES-1,PCR if error detected
01790  5093 BD   9BDB              JSR    MESSAG
01800  5096 A6   21                LDA    1,Y      load error code
01810  5098 BD   AC3D              JSR    A16OUT   error code output
01820  509B 39                     RTS
01830                       *
01831                       * messages
01832                       *
01840            0D0A       CRLF   EQU    $0D0A
01850  509C      0D0A       MES1   FDB    CRLF
01860  509E      44                FCC    'DRIVE NO. ='
01870  50A9      00                FCB    0
01880  50AA      0D0A       MES2   FDB    CRLF
01890  50AC      54                FCC    'TRACK NO. ='
01900  50B7      00                FCB    0
01910  50B8      0D0A       MES3   FDB    CRLF
01920  50BA      53                FCC    'SECTOR NO.='
01930  50C5      00                FCB    0
01940  50C6      0D0A       MES4   FDB    CRLF
01950  50C8      57                FCC    'WRITE(w) or READ(r) ? '
01960  50DE      00                FCB    0
01970  50DF      0D0A       ERRMES FDB    CRLF
01980  50E1      07                FCB    $07
01990  50E2      2A                FCC    '*** ERROR no.=$'
02000  50F1      00                FCB    0
02010                       *
02011                       * working area
02012                       *
02020  50F2      0008       RCB    RMB    8
02030            6000       BUFFER EQU    $6000
02040            5000              END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50F9
PROGRAM ENTRY ADDR=5000
```

```
100 '*
110 '* disk read & write
120 '*
130 DIM D(15)
140 PRINT" DISK READ & WRITE"
150 EXEC &H5000
160 PRINT"      +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F        CHARACTER
170 FOR ADDR=&H6000 TO &H60FF STEP 16
180   PRINT:PRINT RIGHT$("0"+HEX$(ADDR-&H6000),2);"--";
190   FOR BYTE=0 TO 15
200     D(BYTE)=PEEK(ADDR+BYTE)
210   NEXT
220   FOR BYTE=0 TO 15
230     PRINT " ";RIGHT$("0"+HEX$(D(BYTE)),2);
240   NEXT
250   PRINT "       ";
260   FOR BYTE=0 TO 15
```

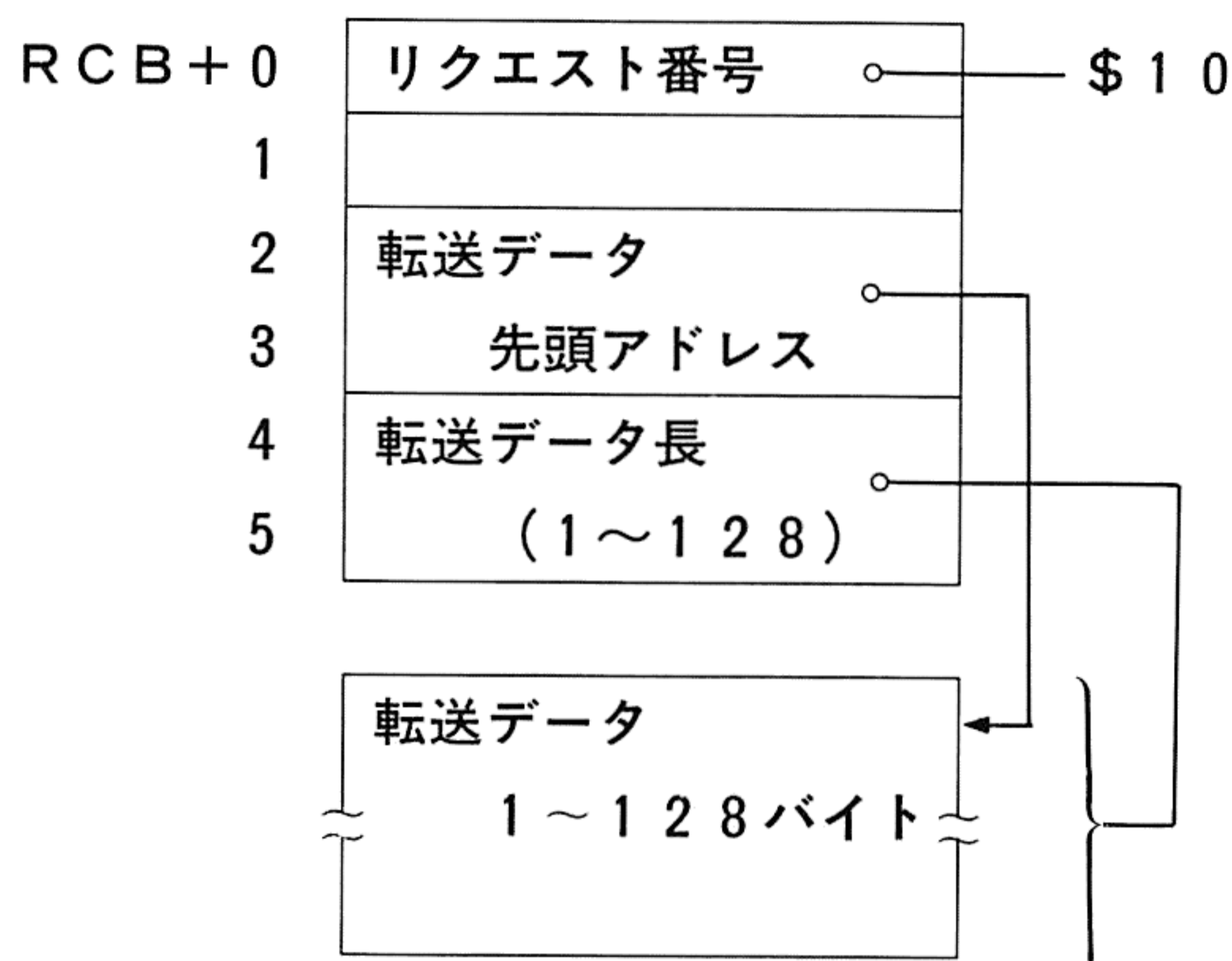
16進表現で 0~F (1桁)の時
フォーマットが崩れぬようにしている.

```
270     IF D(BYTE)<32 OR D(BYTE)=&H7F THEN PRINT "."; ELSE PRINT CHR$(D(BYTE));
280   NEXT
290 NEXT

RUN
 DISK READ & WRITE

DRIVE NO. =0
TRACK NO. =1
SECTOR NO.=4
WRITE(w) or READ(r) ? r
      +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F       CHARACTER

00--- 44 46 4D 43 44 20 20 20 00 00 00 02 00 00 00 00      DFMCD   ........
10--- 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ................
20--- 4D 43 4F 50 59 20 20 20 00 00 00 02 00 00 01 00      MCOPY   ........
30--- 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ................
40--- 53 59 53 44 53 4B 20 20 00 00 00 00 00 00 02 00      SYSDSK  ........
50--- 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ................
60--- 56 4F 4C 43 4F 50 59 20 00 00 00 00 00 00 03 00      VOLCOPY ........
70--- 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ................
80--- 41 55 54 4F 55 54 59 20 00 00 00 00 00 00 04 00      AUTOUTY ........
90--- 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ................
A0--- 50 46 44 45 46 20 20 20 00 00 00 00 00 00 05 00      PFDEF   ........
B0--- 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ................
C0--- 44 45 4D 4F 31 20 20 20 00 00 00 00 00 00 07 00      DEMO1   ........
D0--- 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ................
E0--- 44 45 4D 4F 32 20 20 20 00 00 00 00 00 00 0E 00      DEMO2   ........
F0--- 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ................
Ready
```

BIOS：SUBOUT　ーサブシステム・アウトプットー

機　能　ディスプレイ・サブシステムに対して出力を行います．

パラメータ　　Ｘレジスタ←ＲＣＢ先頭アドレス

ＲＣＢ＋０	リクエスト番号	＄１０
１		
２	転送データ	
３	先頭アドレス	
４	転送データ長	
５	（１～１２８）	

転送データ
１～１２８バイト

復帰情報　ＲＣＢの設定が正しくない（データ長が０，１２９以上など）ときはＲＣＢ＋１番地にＲＣＢエラー（＄０１）がセットされます．

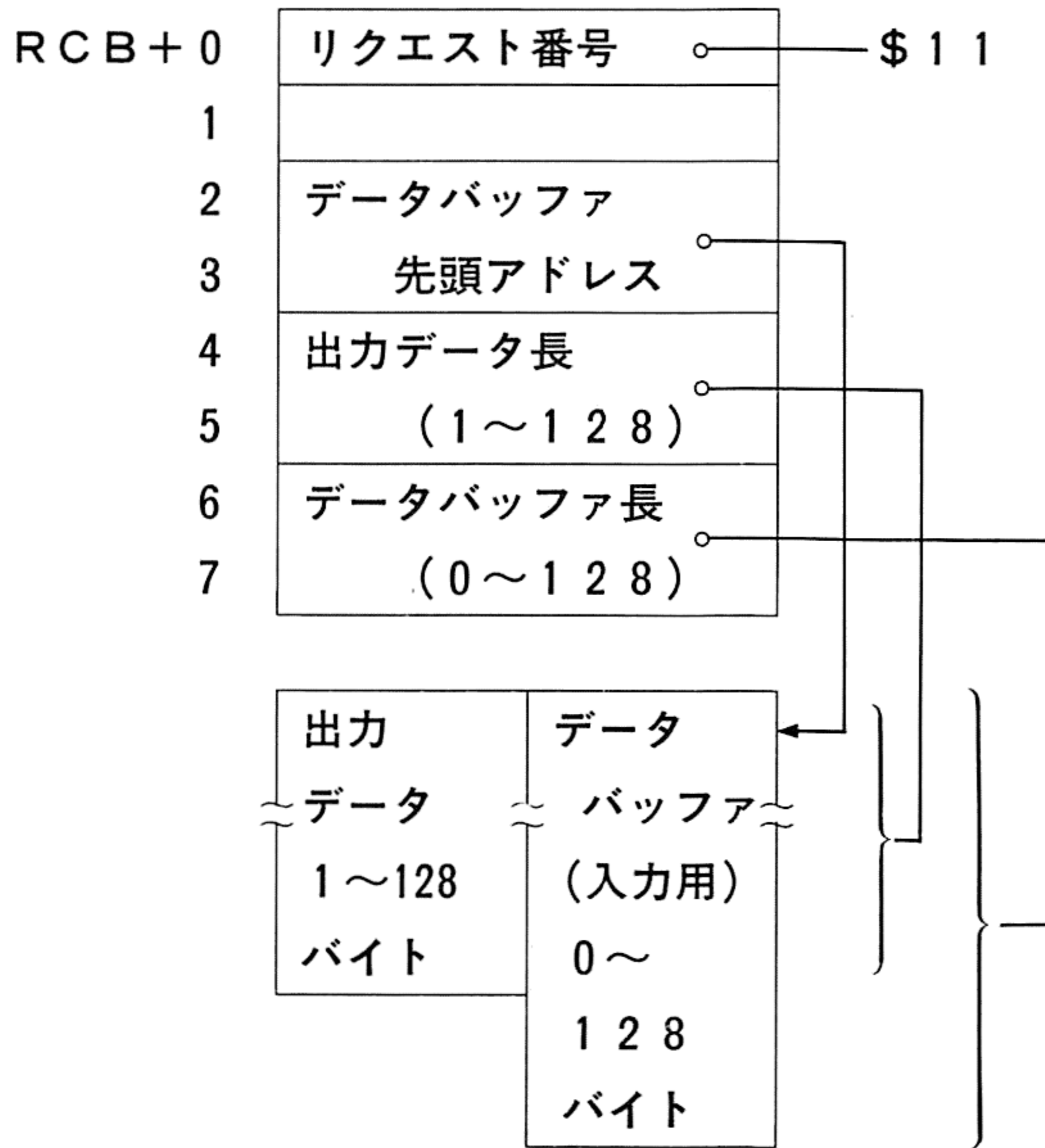
解　説　ディスプレイ・サブシステムに対してコマンド・データ等を出力します．具体的には，サブＣＰＵにＨＡＬＴをかけ，共有ＲＡＭにコマンド・データを転送，ＨＡＬＴを解除します．転送データの内容についてはサブシステムの章を御参照下さい．このリクエストはサブシステムの動作完了を待たないのでサブシステムで生じたエラーについてはユーザーに通知しません．よってＲＣＢの設定が正しければ常にエラーは生じません．また，ルーチン内部の一部でＦＩＲＱをマスクしています．

BIOS：SUBIN　－サブシステム・インプット－

機　　能　ディスプレイ・サブシステムに対して出入力を行います．

パラメータ　　Xレジスタ←RCB先頭アドレス

RCB＋0	リクエスト番号	＄11
1		
2	データバッファ	
3	先頭アドレス	
4	出力データ長	
5	（1～128）	
6	データバッファ長	
7	（0～128）	

出力データ	データバッファ（入力用）
1～128バイト	0～128バイト

復帰情報　サブシステム側から返されたデータがデータバッファにデータバッファ長だけ転送されます．返されたデータよりデータバッファ長が短かい場合にはエラーは生じずに，データが失われます．

またサブシステム側でエラーが生じた場合には，RCB＋1番地にエラー番号がセットされます．（エラー番号はサブシステムのそれと同様なので，サブシステムの方を御参照下さい．）

解　　説　サブシステムに対しコマンド，データを出力し，実行結果をサブシステムから入力します．具体的にはSUBOUTと同様の動作の後，サブシステムがコマンドを実行し終るのを待ってHALTをかけ実行結果を共有RAMからデータバッファに転送，HALTを解除します．

このリクエストもSUBOUT同様，一部でFIRQをマスクしています．

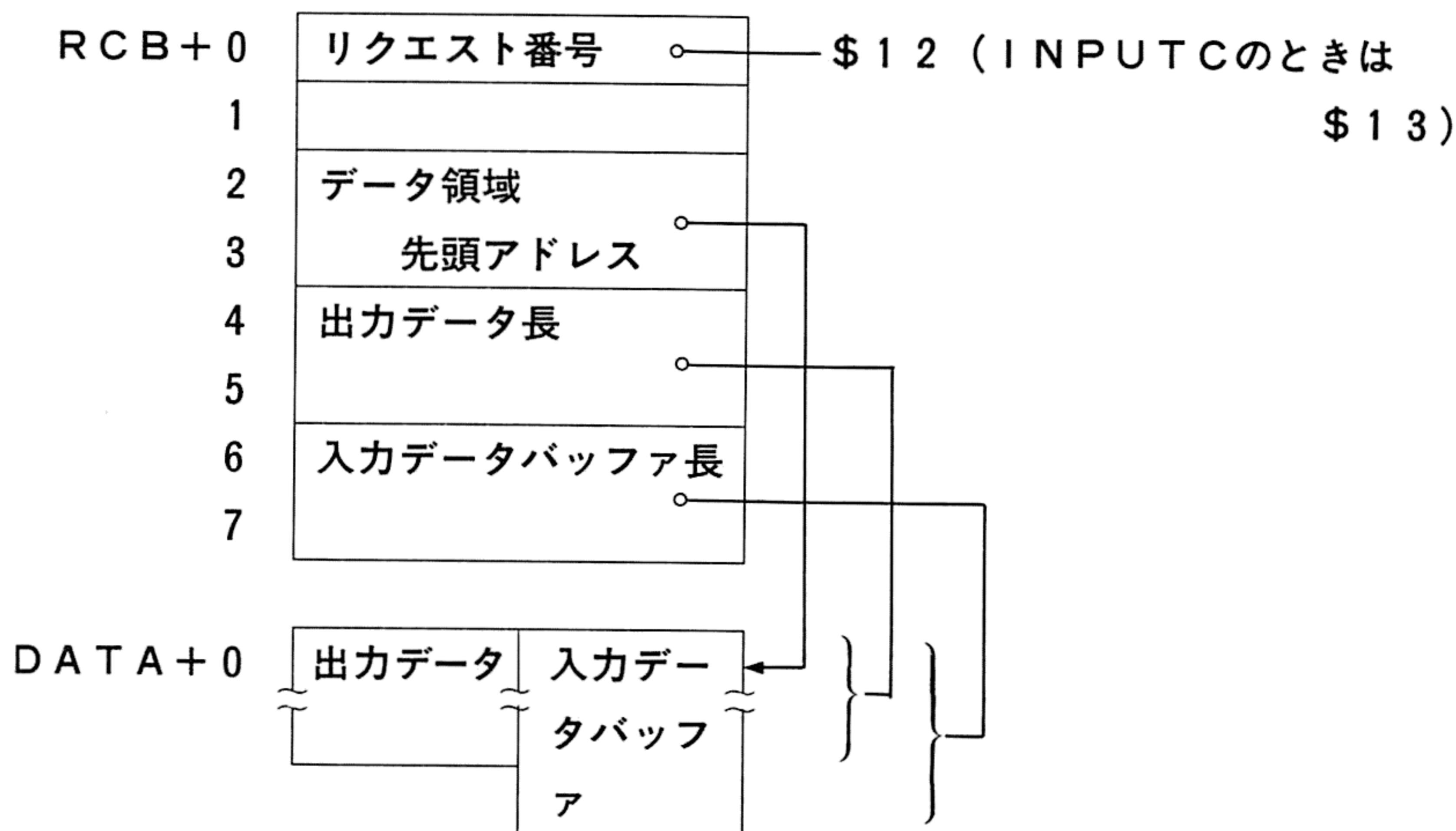
くわしいサブシステムの使用法は，サブシステムの章を御参照下さい．

BIOS：INPUT・INPUTC　－サブシステム１行入力－

機　能　サブシステムから１行入力を行います．

パラメータ　　Ｘレジスタ←ＲＣＢ先頭アドレス

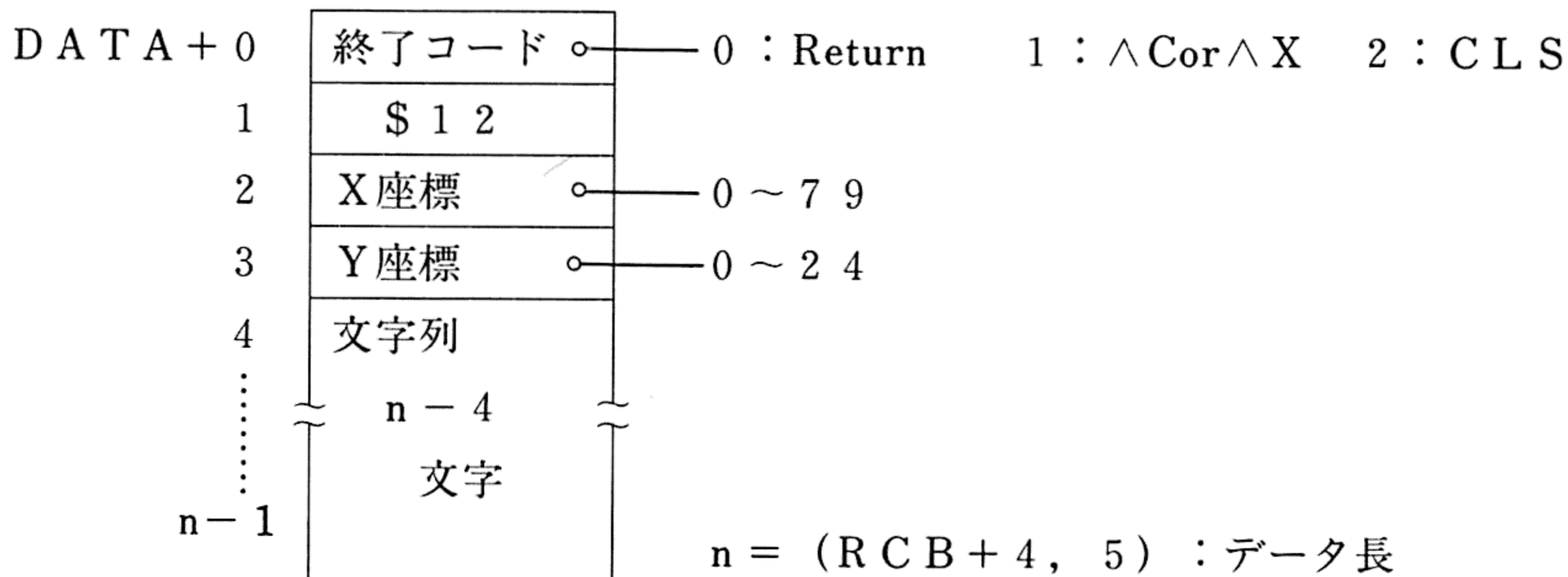
RCB＋0	リクエスト番号	＄１２（ＩＮＰＵＴＣのときは＄１３）
1		
2	データ領域	
3	先頭アドレス	
4	出力データ長	
5		
6	入力データバッファ長	
7		

DATA＋0	出力データ	入力データバッファ

復帰情報　画面上で変更されたフィールドのうち最上行のフィールドを入力データバッファに転送します．バッファ長が足りなかった場合には（ＲＣＢ＋１）がバッファオーバーエラー（＄４Ｆ）を示します．

解　説　画面に出力データを出力した後，キーボードより文字列を入力します．Returnキー（又は∧Ｃ，∧Ｘ，ＣＬＳ）が押されたら入力データバッファに変更されたフィールドを上の方から転送します．変更されたフィールドが複数あった場合にはＩＮＰＵＴＣを実行することによって次のフィールドを取り出す必要があります．

入力データの形式を示します．

DATA＋0	終了コード	０：Return　１：∧Cor∧Ｘ　２：ＣＬＳ
1	＄１２	
2	Ｘ座標	０～７９
3	Ｙ座標	０～２４
4	文字列	
⋮	ｎ－４	
ｎ－１	文字	

ｎ＝（ＲＣＢ＋４，５）：データ長

このリクエストでは常に終了コードを転送するので，すべてのフィールドを転送し終ったかどうかは（RCB+4, 5）が1かどうかで判断します．1でないときは，さらにINPUTCにより変更フィールドを取り出す必要があります．

サンプル　INPUT，INPUTC，OUTPUTを用いたサンプルです．
入力された文字列を大文字にして出力します．

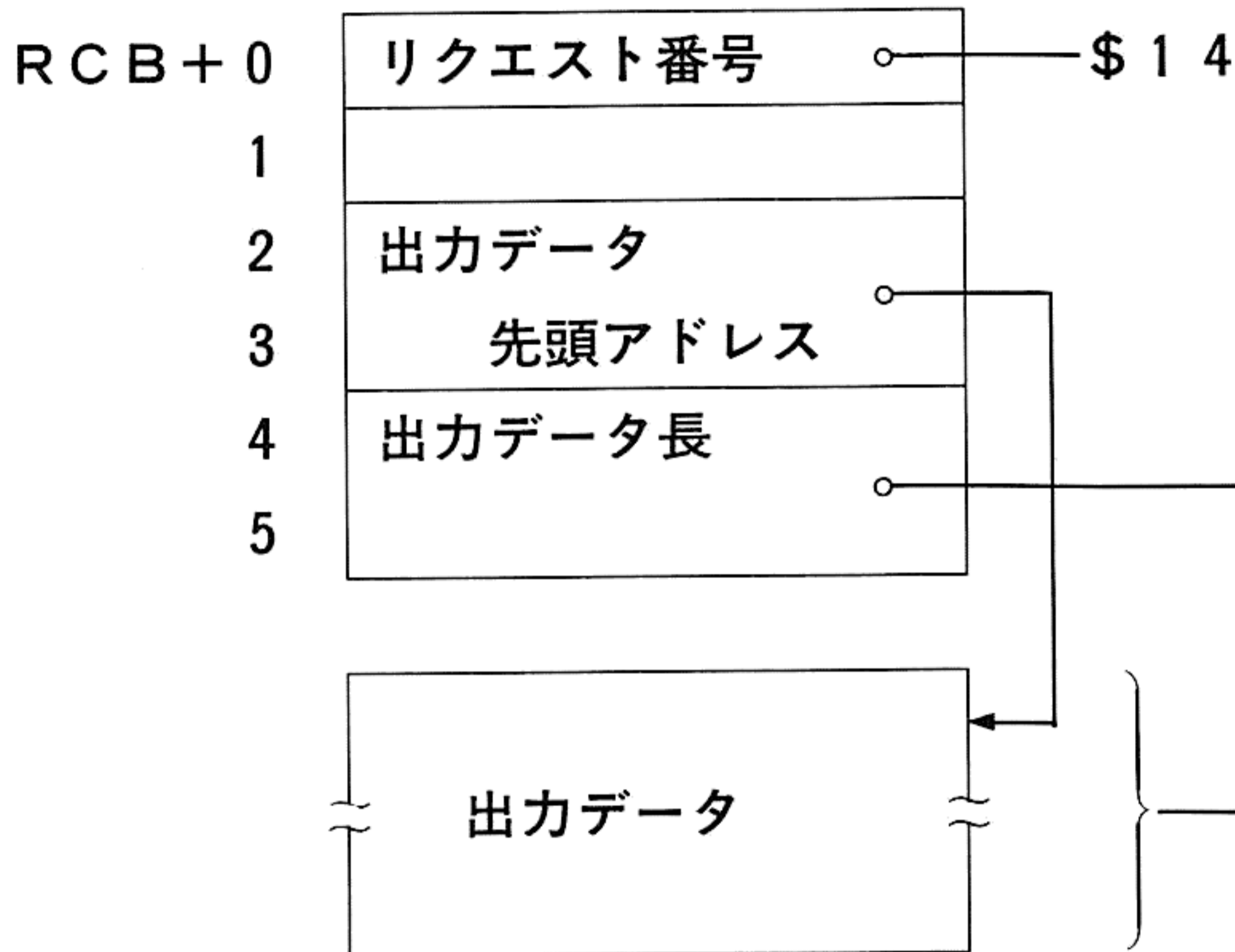
```
PAGE  001  (           ,          )

01000                          *
01010                          * convert string into capitals
01020                          *                 (BIOS:INPUT,INPUTC,OUTPUT)
01030                                 OPT    M,NOS,P=255,NOO,NOG
01040               FBFA       BIOS   EQU    $FBFA
01050                          *
01060  5000                           ORG    $5000
01070               5000       START  EQU    *
01080                          *
01090                          * put prompt message
01100                          *
01110  5000 31      8D 010E           LEAY   DATA,PCR input data store area
01120  5004 30      8D 00A6           LEAX   PROMPT,PCR load rcb for output
01130  5008 AD      9F FBFA           JSR    [BIOS]   output prompt
01140                          *
01150                          * get string
01160                          *
01170  500C 30      8D 00D2           LEAX   RCB,PCR load rcb address
01180  5010 33      8D 00D6           LEAU   BUFFER,PCR buffer for INPUT
01190  5014 EF      02                STU    2,X
01200  5016 CC      1107              LDD    #$1107   set field & color=7
01210  5019 ED      C4                STD    ,U
01220  501B CC      0002              LDD    #2       set data length=2
01230  501E ED      04                STD    4,X
01240  5020 86      12                LDA    #$12     bios request INPUT
01250  5022 A7      84         CONT   STA    ,X       set request no.
01260  5024 CC      0028              LDD    #40      set buffer length
01270  5027 ED      06                STD    6,X
01280  5029 AD      9F FBFA           JSR    [BIOS]   input characters
01290  502D 25      64    5093        BCS    ERROR    if error detected
01300  502F EC      04                LDD    4,X      get data length
01310  5031 1083    0001              CMPD   #1       input data length=1 ?
01320  5035 27      20    5057        BEQ    INEND    yes,all field end
01330  5037 E6      C4                LDB    ,U       get end key flag
01340  5039 26      1C    5057        BNE    INEND    if not end by return key
01350  503B 34      40                PSHS   U        save buffer address
01360  503D 33      44                LEAU   4,U      skip 4 chr
01370                          *                (endkey_flag,$12,x,y)
01380  503F E6      05                LDB    5,X      get inputline length
01390                          *                (length<40)
01400  5041 C0      04                SUBB   #4       skip 4 chr
01410  5043 23      07    504C MOVE   BLS    NEXT     if chr end
01420  5045 A6      C0                LDA    ,U+      transfer
01430  5047 A7      A0                STA    ,Y+      inputline to data area
01440  5049 5A                        DECB
01450  504A 20      F7    5043        BRA    MOVE
01460  504C CC      0D0A       NEXT   LDD    #$0D0A   add CR & LF
01470  504F ED      A1                STD    ,Y++
01480  5051 35      40                PULS   U        buffer address come back
01490  5053 86      13                LDA    #$13     bios request INPUTC
01500  5055 20      CB    5022        BRA    CONT     get next field
01510                          *
01520                          * convert string into capitals
01530                          *
01540  5057 33      8D 00B7    INEND  LEAU   DATA,PCR convert to capitals
01550  505B A6      C4         INE01  LDA    ,U       load character
01560  505D 81      61                CMPA   #'a      test 'a'..'z'
01570  505F 25      08    5069        BLO    CAP
01580  5061 81      7A                CMPA   #'z
01590  5063 22      04    5069        BHI    CAP
```

```
01600  5065 80   20                SUBA   #'a-'A   if 'a'-'z'
01610  5067 A7   C4                STA    ,U
01620  5069 33   41          CAP   LEAU   1,U
01630  506B 34   40                PSHS   U        get Yreg-Ureg
01640  506D 1F   20                TFR    Y,D
01650  506F A3   E1                SUBD   ,S++
01660  5071 26   E8    505B        BNE    INE01    if not end
01670                        *
01680                        * put converted strings
01690                        *
01700  5073 33   8D 0054     OUTPUT LEAU  OUTMES,PCR output message
01710  5077 CC   0017              LDD    #.OUTME-OUTMES
01720  507A 8D   0A    5086        BSR    OUT01
01730  507C 33   8D 0092           LEAU   DATA,PCR load data top addr.
01740  5080 34   40                PSHS   U        get Yreg-Ureg
01750  5082 1F   20                TFR    Y,D
01760  5084 A3   E1                SUBD   ,S++
01770  5086 ED   04          OUT01 STD    4,X      set data length
01780  5088 EF   02                STU    2,X      set data top address
01790  508A 86   14                LDA    #$14     bios request OUTPUT
01800  508C A7   84                STA    ,X       set request no.
01810  508E AD   9F FBFA           JSR    [BIOS]   output chatacters
01820  5092 39                     RTS
01830                        *
01840                        * error message output
01850                        *
01860  5093 33   8D 0005     ERROR LEAU   ERRMES,PCR
01870  5097 CC   0012              LDD    #.ERRME-ERRMES
01880  509A 20   EA    5086        BRA    OUT01
01890  509C      0D0A        ERRMES FDB   $0D0A
01900  509E      20                FCC    ' BIOS ERROR !!! '
01910            50AE        .ERRME EQU   *
01920                        *
01930                        * messages
01940                        *
01950  50AE      1400        PROMPT FDB   $1400    bios request OUTPUT
01960  50B0      50B4              FDB    PROMP1
01970  50B2      0017              FDB    .PROMP-PROMP1
01980  50B4      0D0A        PROMP1 FDB   $0D0A
01990  50B6      20                FCC    ' please input -------'
02000            50CB        .PROMP EQU   *
02010                        *
02020  50CB      0D0A        OUTMES FDB   $0D0A
02030  50CD      20                FCC    ' input characters ==='
02040            50E2        .OUTME EQU   *
02050                        *
02060                        * working area
02070  50E2      0008        RCB   RMB    8        rcb ( 8 bytes full use)
02080  50EA      0028        BUFFER RMB   40       buffer for bios
02090            5112        DATA  EQU    *        data store area top addr.
02100            5000              END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5111
PROGRAM ENTRY ADDR=5000


exec &H5000

 please input -------This is a pen.
 input characters ===THIS IS A PEN.

Ready
exec &h5000

 please input -------string over 40 characters --> bios error !!!!
 BIOS ERROR !!!
Ready
exec

 please input -------Shuwa system trading co.,ltd.
 input characters ===SHUWA SYSTEM TRADING CO.,LTD.

Ready
```

BIOS：OUTPUT　―サブシステム・キャラクタデータ出力―

機　能　指定された文字列を画面に出力します．

パラメータ　　Ｘレジスタ←ＲＣＢ先頭アドレス

RCB+	内容	
0	リクエスト番号	＄１４
1		
2	出力データ	
3	先頭アドレス	
4	出力データ長	
5		

出力データ

復帰情報　（ＲＣＢ＋１）：エラー番号（サブシステムの章を参照）

解　説　サブシステムにコマンドを送ることによって文字列を画面に出力します．出力データ長が１２４より大きいときは，ルーチンが分割してサブシステムに転送します．

サンプル　ＩＮＰＵＴの項を御参照下さい．

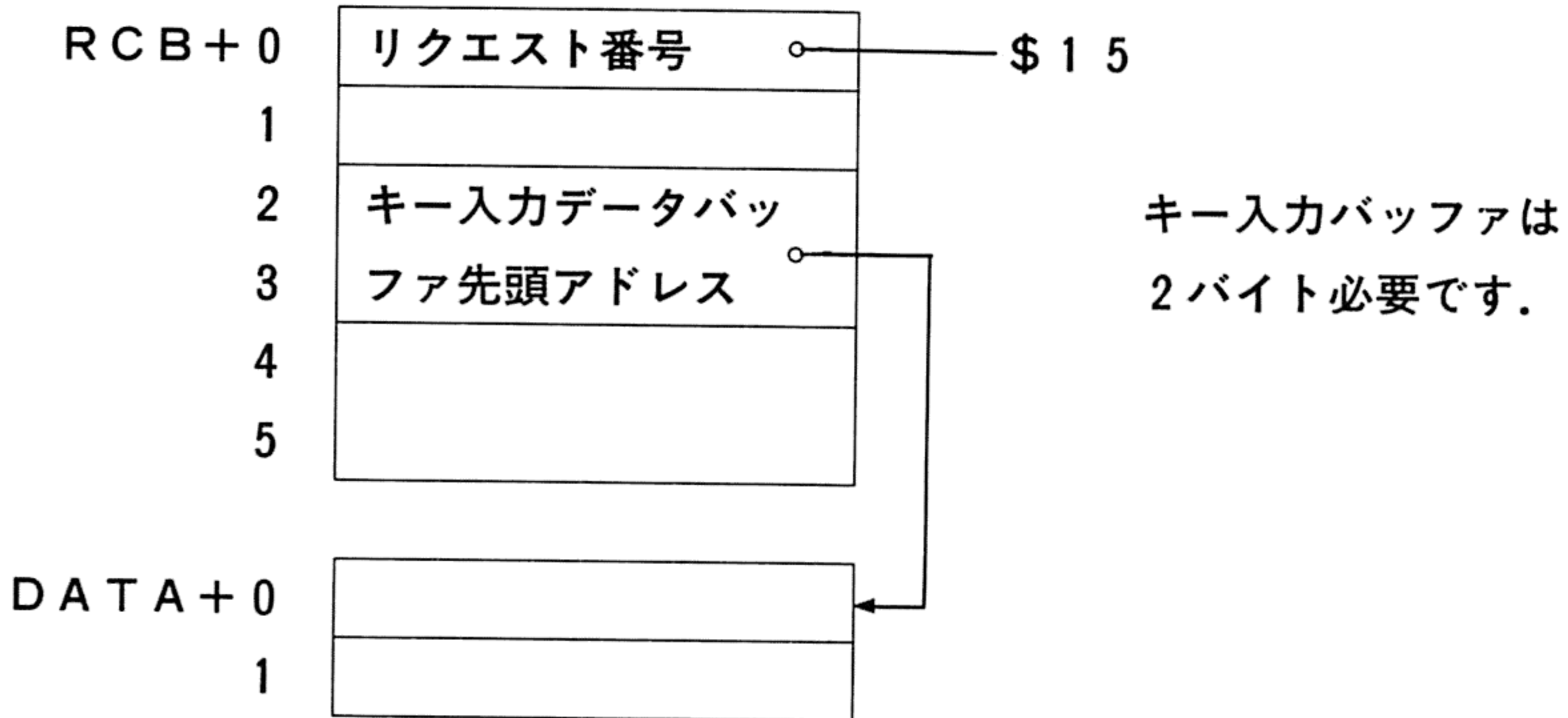
BIOS：KEYIN　－キーボード１文字入力－

機　能　サブシステムからキーボードの情報をうけとります．

パラメータ　Ｘレジスタ←ＲＣＢ先頭アドレス

ＲＣＢ＋０	リクエスト番号	＄１５
１		
２	キー入力データバッファ先頭アドレス	
３		
４		
５		

ＤＡＴＡ＋０	
１	

キー入力バッファは２バイト必要です．

このリクエストはＲＣＢ領域として６バイトしか使用しないので，キー入力データバッファとしてＲＣＢ＋６，ＲＣＢ＋７の２バイトの使用も可能です．

復帰情報　（ＤＡＴＡ＋０）：文字コード
（ＤＡＴＡ＋１）：キー入力フラグ
０：入力なし
１：入力あり

また，ＲＣＢ＋４，５番地にはデータバッファの長さがセットされるので常に（ＲＣＢ＋４，５）は２となります．

解　説　キー入力の状態を調べます．しかし，サブシステムがキーの先行入力を可能にしている場合には先行入力されたものが返されるので注意して下さい．

サンプル　入力したキーのキーコードを表示します．

```
PAGE   001 (          ,          )

01000                          *
01010                          * KEY INPUT & DISPLAY KEY CODE
01020                          *                 (BIOS:KEYIN_)
01030                                  OPT     M,NOS,P=255,NOO,NOG
01040                9BDB'     MESSAG  EQU     $9BDB    output from (X+1) till 0
01050                FBFA      BIOS    EQU     $FBFA    bios (extend-indirect)
01060                AC3D      A16OUT  EQU     $AC3D    output A reg in HEX
01070                D678      ABORTT  EQU     $D678    BASIC Abort test
01080                9B50      CRLF    EQU     $9B50    CR & LF
01090                          *
01100    5000                          ORG     $5000
01110                          *
01120                5000      START   EQU     *
01130    5000 BD     9B50              JSR     CRLF
01140                          *
```

```
01150                       * get key code
01160                       *
01170  5003 30   8D 0034    NEXT    LEAX    RCB,PCR
01180  5007 86   15                 LDA     #$15      bios request KEYIN
01190  5009 A7   84                 STA     ,X
01200  500B 33   06                 LEAU    6,X       get buffer address
01210  500D EF   02                 STU     2,X       set buffer address
01220  500F AD   9F FBFA    LOOP    JSR     [BIOS]    call bios
01230  5013 A6   41                 LDA     1,U       get keyin flag
01240  5015 27   F8   500F          BEQ     LOOP      if no keyin
01250  5017 A6   C4                 LDA     ,U        get keycode
01260                       *
01270                       * output key code in HEX
01280                       *
01290  5019 34   02                 PSHS    A
01300  501B 30   8D 000C            LEAX    MES1-1,PCR message output
01310  501F BD   9BDB               JSR     MESSAG
01320  5022 35   02                 PULS    A
01330  5024 BD   AC3D               JSR     A16OUT    output keycode in HEX
01340  5027 BD   D678               JSR     ABORTT    BREAK key test
01350  502A 20   D7   5003          BRA     NEXT
01360                       *
01370                       * message
01380                       *
01390  502C      20         MES1    FCC     '    Key code =$' output message
01400  503A      00                 FCB     0
01410                       *
01420                       * working area
01430                       *
01440  503B      0006       RCB     RMB     6         only needs 6 bytes for rcb
01450  5041      0002               RMB     2         for buffer
01460            5000               END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5042
PROGRAM ENTRY ADDR=5000


exec &H5000

    Key code =$2A    Key code =$2F    Key code =$2B    Key code =$2D    Key code =$37
    Key code =$38    Key code =$39    Key code =$3D    Key code =$34    Key code =$35
    Key code =$36    Key code =$2C    Key code =$31    Key code =$32    Key code =$33
    Key code =$0D    Key code =$30    Key code =$2E    Key code =$20

Abort
Ready
```

2. Display Sub System

2－1　Display　Sub　Systemの概略

ＦＭ７は２つのＣＰＵ（正確には，キーボートコントロール用４ビットワンチップマイコンＭＢ８８４０１をふくんで３つ）をもっています．このことによりメインＣＰＵは演算に，サブＣＰＵはキーボード管理と画面表示に専念できるように設計されています．

メインＣＰＵが画面表示の必要を生じた時には，共有ＲＡＭ（メインＣＰＵの＄ＦＣ８０～＄ＦＣＦＦ，サブＣＰＵの＄Ｄ３８０～＄Ｄ３ＦＦ）を介して，ディスプレイサブシステムと称するサブＣＰＵにコマンドを送り表示を行います．

一方サブＣＰＵの側では，共有ＲＡＭにセットされるコマンドを逐次実行し，データをメインＣＰＵに返す必要のある場合は，同じく共有ＲＡＭを通してメインＣＰＵにデータを返します．

このように，ディスプレイサブシステム（サブＣＰＵ）は一種のインテリジェントグラフィック端末のような存在であるわけです．

メインＣＰＵとサブＣＰＵの間でのデータのやりとりは，ＨＡＬＴ，ＢＵＳＹなどのインタフェース信号線を操作することによって行われますが，ユーザーはＢＩＯＳに用意されているＳＵＢＯＵＴ・ＳＵＢＩＮのリクエストを使用することにより，各種信号線の操作をＢＩＯＳにまかせ，信号線の操作を意識せずにサブＣＰＵとデータのやりとりを行うことができます．

このＢＩＯＳを使用した使用法については，ＢＩＯＳの項及び各コマンドのサンプル等を御参照下さい．

もう一つの使用法として，直接信号線を操作してサブＣＰＵとやりとりを行う方法があります．

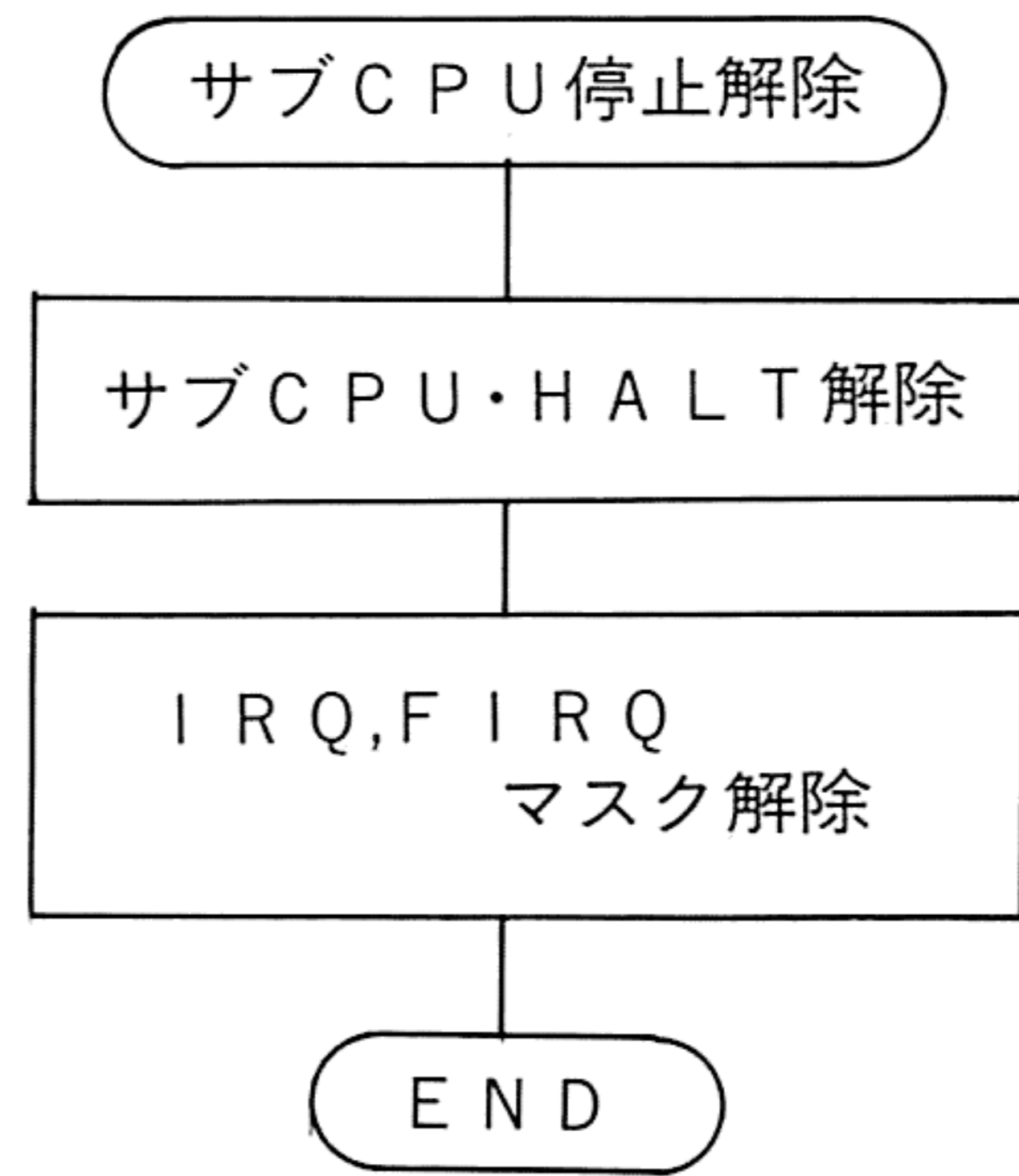
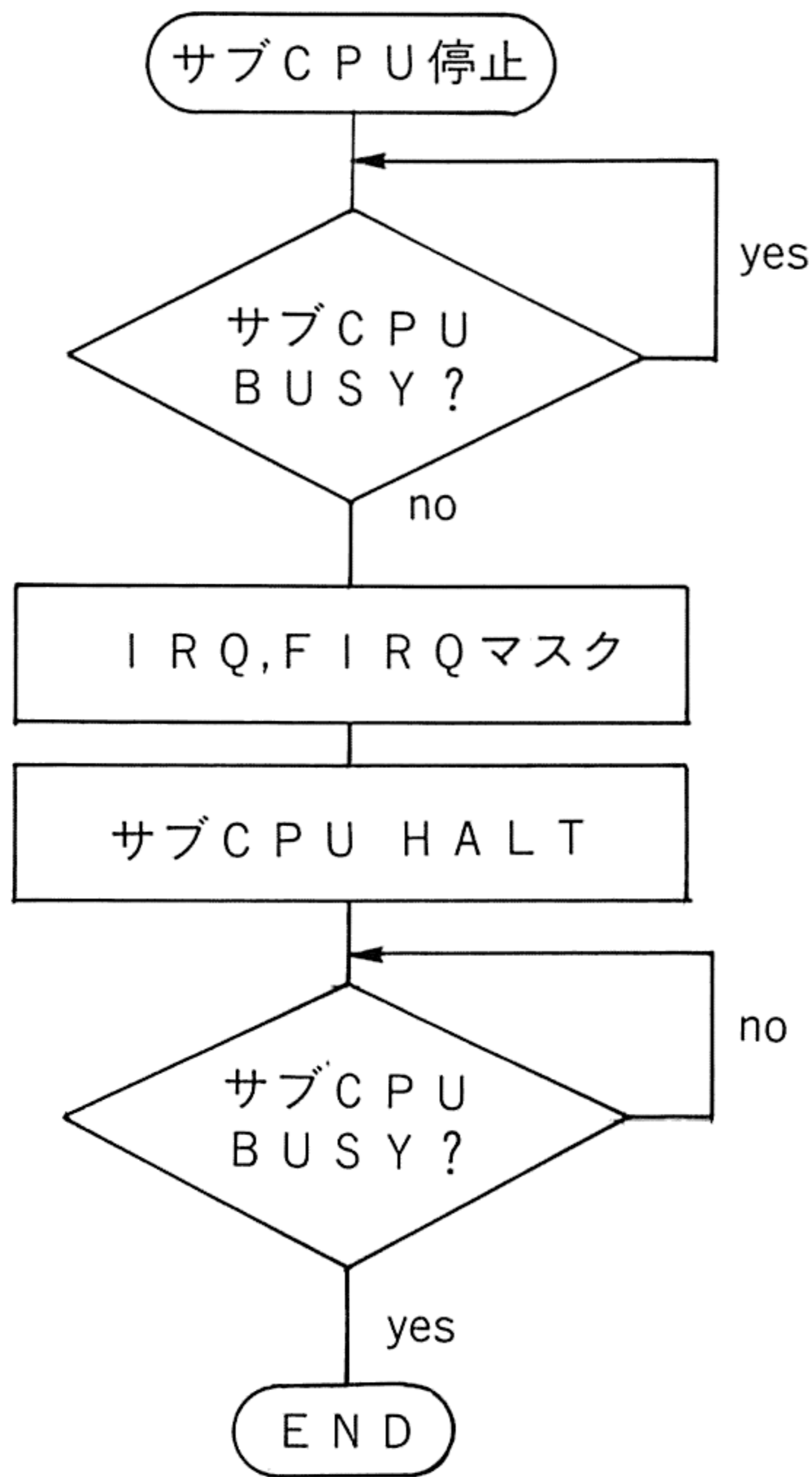
通常，共有ＲＡＭはサブＣＰＵのアドレスバスに接続されていて，メインＣＰＵからはアクセスできません．そこでまず，サブＣＰＵを停止させて共有ＲＡＭをメインＣＰＵにあけ渡させ，共有ＲＡＭにコマンドをセットします．コマンドをセットし終りしだい，サブＣＰＵの停止を解除し，サブＣＰＵにコマンドを実行させます．

くわしい使用法は，図とサンプルプログラムを御参照下さい．

コンソールの機能・フィールドの概念・アトリビュート文字などについては，「ユーザーズマニュアル・システム仕様」を御参照下さい．

サブCPU停止

サブCPU BUSY?
yes
no
IRQ,FIRQマスク
サブCPU HALT
サブCPU BUSY?
no
yes
END

サブCPU停止解除
サブCPU・HALT解除
IRQ,FIRQ マスク解除
END

サブCPU停止解除の後すぐにサブCPUを停止する場合は、Dilayをかけて下さい．(少なくともBIOS内では、この処理が行われています)．

サンプル サブCPUを直接(BIOSを使わずに)動作させます．

```
PAGE   001  (          ,          )

01000                          *
01010                          * SUBSYSTEM DIRECT CONTROL
01020                          *
01030                                 OPT    M,NOS,NOG,NOO,PAGE=255
01040               9BDB       MESSAG EQU    $9BDB    output from (X+1) till 0
01050               AC3D       A16OUT EQU    $AC3D    output A reg in HEX
01060                          *
01070    5000                         ORG    $5000
01080               5000       START  EQU    *
01090                          *
01100                          * subcpu halt & command set
01110                          *
01120    5000 8D    52    5054        BSR    HALT     subCPU HALT
01130    5002 30    8D 007C           LEAX   DATA,PCR
01140    5006 108E  FC80              LDY    #$FC80
01150    500A E6    8D 0073           LDB    LENGTH,PCR
01160    500E A6    80         LOOP   LDA    ,X+      transfer to shared RAM
01170    5010 A7    A0                STA    ,Y+
01180    5012 5A                      DECB
01190    5013 26    F9    500E        BNE    LOOP
01200                          *
01210                          * subcpu halt reset & wait a moment
01220                          *
```

```
01230  5015 8D    53   506A         BSR     HRESET   subCPU HALT reset
01240                         * if HALT as soon as HRESET,wait a moment !
01250  5017 86    18                LDA     #24      dilay about 50 micro sec
01260  5019 4A                DILAY DECA
01270  501A 26    FD   5019         BNE     DILAY
01280                         *
01290                         * subcpu halt & read error code
01300                         *
01310  501C 8D    36   5054         BSR     HALT     subCPU HALT
01320  501E B6    FC80              LDA     $FC80    get status
01330  5021 8D    52   5075         BSR     REDYRQ   not command set
01340  5023 8D    45   506A         BSR     HRESET   subCPU HALT reset
01350  5025 84    7F                ANDA    #$7F     get error no.
01360  5027 27    0E   5037         BEQ     NOERR
01370  5029 34    02                PSHS    A        error detected
01380  502B 30    8D 0008           LEAX    ERRMES-1,PCR
01390  502F BD    9BDB              JSR     MESSAG
01400  5032 35    02                PULS    A
01410  5034 BD    AC3D              JSR     A16OUT   output error no.
01420  5037 39                NOERR RTS
01430                         *
01440                         * error message
01450                         *
01460  5038       0D0A        ERRMES FDB    $0D0A
01470  503A       2A                FCC     '*** Subsystem Error NO.=$'
01480  5053       00                FCB     0
01490                         *
01500                         * subroutine:SUBCPU HALT
01510                         *
01520  5054 34    02          HALT  PSHS    A
01530  5056 B6    FD05        BUSY  LDA     $FD05
01540  5059 2B    FB   5056         BMI     BUSY     wait until ready
01550  505B 1A    50                ORCC    #$50     IRQ,FIRQ MASK
01560  505D 86    80                LDA     #$80
01570  505F B7    FD05              STA     $FD05    HALT subCPU
01580                         *                wait acknowledgement
01590  5062 B6    FD05        READY LDA     $FD05
01600  5065 2A    FB   5062         BPL     READY    wait until BUSYflag set
01610  5067 35    02                PULS    A
01620  5069 39                      RTS                    PULS  PC, A
01630                         *
01640                         * subroutine:SUBCPU HALT RESET
01650                         *
01660  506A 34    02          HRESET PSHS   A                  HRESET  CLR   $FD05
01670  506C 4F                      CLRA                                   #$AF
01680  506D B7    FD05              STA     $FD05    subCPU halt reset  ANDC
01690  5070 1C    AF                ANDCC   #$AF     IRQ,FIRQ mask reset RTS
01700  5072 35    02                PULS    A
01710  5074 39                      RTS
01720                         *
01730                         * subroutine:set ready request
01740                         *  if not set command then set 'ready request'
01750                         *
01760  5075 34    02          REDYRQ PSHS   A
01770  5077 B6    FC80              LDA     $FC80
01780  507A 8A    80                ORA     #$80     MSB on
01790  507C B7    FC80              STA     $FC80
01800  507F 35    82                PULS    A,PC
01810                         *
01820                         * subsystem command
01830                         *
01840  5081       13          LENGTH FCB    .DATA-DATA
01850  5082       0000        DATA  FDB     0
01860  5084       19                FCB     $19   SIMBOL
01870  5085       05                FCB     5
01880  5086       00                FCB     0
01890  5087       00                FCB     0
01900  5088       06                FCB     6
01910  5089       06                FCB     6
01920  508A       0064              FDB     100
01930  508C       0032              FDB     50
01940  508E       06                FCB     .STR-STR
01950  508F       46          STR   FCC     'FM-7 !'
01960             5095        .STR  EQU     *
```

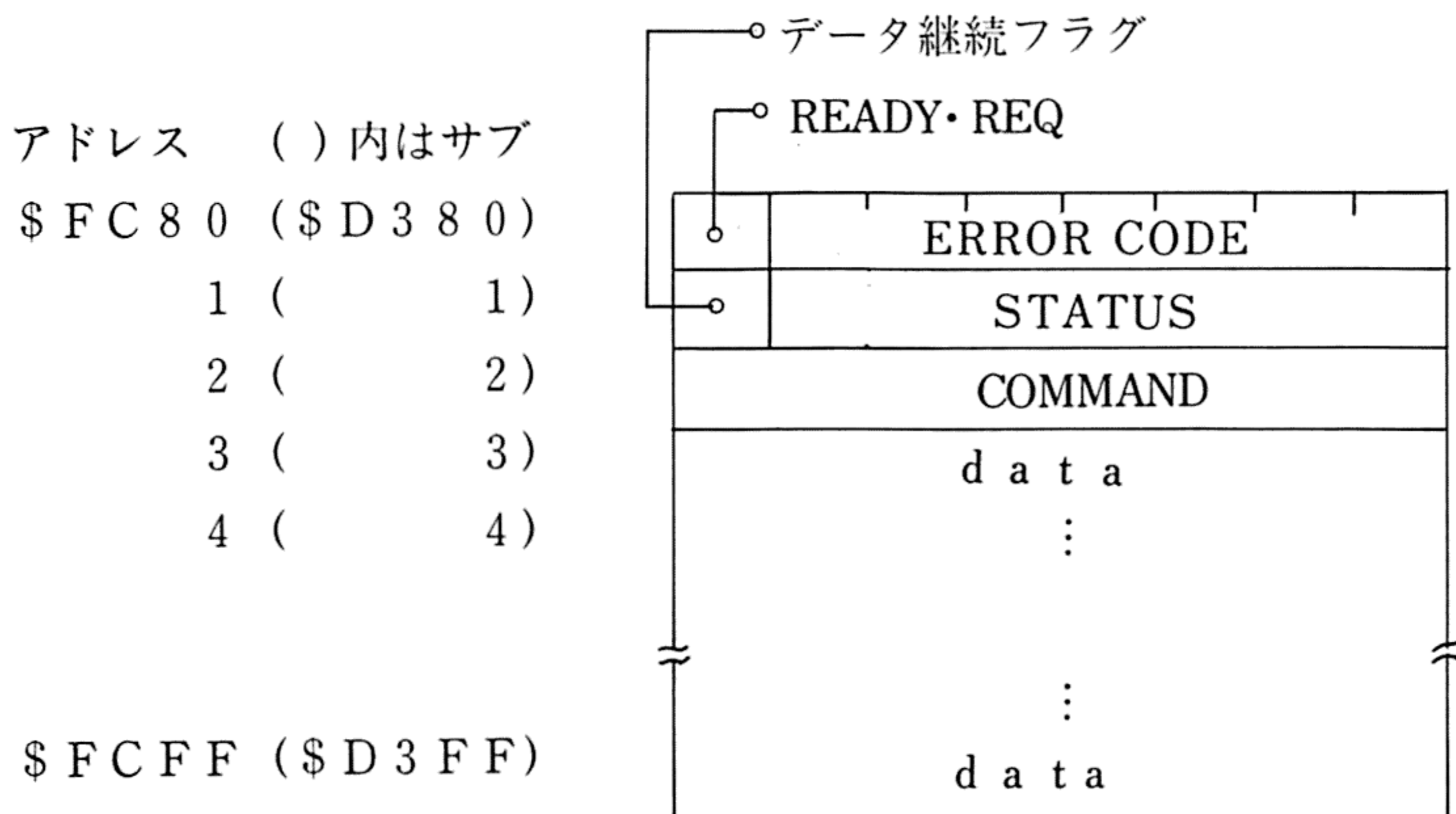
```
01970              5095      .DATA  EQU    *
01980              5000             END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5094
PROGRAM ENTRY ADDR=5000
```

```
exec &H5000

Ready
hardc 2
```

FM-7 !

〈共有RAMのフォーマット〉

サブCPUとメインCPUの情報交換の場である共有RAMのフォーマットについて述べます．

アドレス	( )内はサブ	
$FC80	($D380)	ERROR CODE（bit7: READY・REQ）
1	(1)	STATUS（bit7: データ継続フラグ）
2	(2)	COMMAND
3	(3)	data
4	(4)	⋮
$FCFF	($D3FF)	data

●READY REQ（$FC80 bit7）

サブCPUをエラーコード読み取りなどのためにHALTし，新しくコマンドを設定しない場合に，1にする必要があります．このことにより，サブCPUはコマンドを実行せず，Readyの状態になります．

●ERROR CODE（$FC80 bit0～6）

コマンド実行中にエラーが生じた時，エラーコードがセットされます．正常に実行を終了したときには0がセットされます．

●データ継続フラグ（$FC81 bit7）

コマンドを送るときに，それに付属するデータを一度に送りきれない場合は，このフラグを1にしてコマンドを設定します．これによりサブシステムはまだデータが続くことを知り，次のデータをまちます．残りのデータはコマンドCONTにより転送しますが，データのフォーマットについては各コマンドを参照して下さい．

またサブCPUからデータが返される時にも一度に送りきれない場合は，このフラグが1になっているので，残りをコマンドCONTにより要求します．

●STATUS

サブシステム側から，メインCPUに対して割り込みの必要が生じた場合は，その原因のフラグをONにして，メインCPUにFIRQをかけます．このときメ

インＣＰＵはステータスを調べ，それぞれに応じた処理を行う必要があります．

bit 6　　：０時割り込み
bit 5　　：Interval Timer 割り込み
bit 4　　：Timer 割り込み
bit 0～3　：ＰＦキー割り込み，ＰＦキーコード　１～１０（＄０Ａ）

●ＣＯＭＭＡＮＤ

サブＣＰＵに命令の種類を指定するバイトです．これについては，サブシステムコマンド一覧表を御参照下さい．

サブＣＰＵをＨＡＬＴしたにもかかわらず，コマンドを設定して新しい動作をさせる必要のない時は，ＲＥＡＤＹ　ＲＥＱ　をＯＮにしなければなりません．

●data

サブシステム・コマンドに付随するパラメータやデータ等を入れます．この部分のフォーマットはコマンドごとに異なるので，各コマンドの解説を御参照下さい．

サブシステム・コマンド一覧表

コード 16 進	コマンド名	内　　容	BIOS (注)
0 1	INIT	コンソール初期化	○
0 2	ERASE	画面クリア（文字色指定なし）	○
0 3	PUT	文字列表示	○
0 4	GET	文字列表示・入力	×
0 5	GETC	GETの未転送分を転送	×
0 6	GCBLK1	枠内文字コード読み取り	×
0 7	PCBLK1	枠内文字コード表示	○
0 8	GCBLK2	枠内文字コード及び属性読み取り	×
0 9	PCBLK2	枠内文字コード及び属性表示	○
0 A	GBADR	バッファアドレス読み取り	×
0 B	TABSET	TAB位置設定	○
0 C	CNSCTL	コンソール機能設定	○
0 D	ERASE2	画面クリア（文字色指定あり）	○
1 5	LINE	直線又は四角の表示	○
1 6	CHAIN	連続直線表示	○
1 7	POINT	点表示	○
1 8	PAINT	ペイント	○
1 9	SYMBOL	文字列表示（大きさ・角度付き）	○
1A	CCOLOR	枠内の色を変更	○
1B	GGBLK1	枠内ドット読み取り	×
1C	PGBLK1	枠内ドット表示	○
1D	GGBLK2	枠内ドット読み取り（色付き）	×
1E	PGBLK2	枠内ドット表示（色付き）	○
1F	GCURS	座標読み取り	×
2 0	CLINE	文字による直線又は四角の表示	○

（続く）

注) BIOSの項は，BIOSのSUBOUTで使用することができるか否かを示してあります．　○：SUBOUT，SUBINどちらでも可
×：SUBINのみ可

サブシステム・コマンド一覧表

コード 16進	コマンド名	内　　容	BIOS
29	INKEY	キーコード読み取り	×
2A	DEFPF	PFキーに文字列を定義	○
2B	GETPF	PFキー定義文字列読み取り	×
2C	INTCTL	PFキー割り込み選択	○
3D	SETIME	タイマセット	○
3E	GETIME	タイマ読み取り	×
3F	TEST	メンテナンスコマンド	**
64	CONT	未転送データ転送	*

*：このコマンドを使用する前のコマンドによる.

**：メンテナンスコマンドの用途による.

サブシステム・エラーコード一覧表

エラーコード（16進）	
3C	INITコマンド・パラメータエラー
3D	コンソール座標エラー
3E	オーダーエラー
3F	グラフィック座標エラー
40	ファンクションコードエラー
41	座標数エラー
42	文字数エラー
43	色数エラー
44	PF番号エラー
45	コマンド・パラメータエラー
46	コマンドコードエラー

注）このエラーコードはBIOS経由で使用した時には，そのままBIOSのエラーコードとなります.

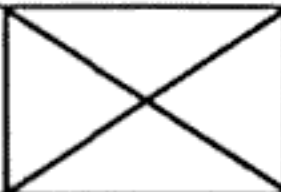
オーダーシーケンス一覧表

ＰＵＴ・ＧＥＴコマンド時におけるオーダーの一覧表です．この表に出ていない＄００～＄１Ｆまでのコードは全て動作しません．

オーダー名	コード1　コード2　コード3	機　　能
ＥＬ	＄０５	フィールド終りまでの文字を消去
ＢＥＬ	＄０７	ベルをならします
ＢＳ	＄０８	バッファアドレスを１つ戻す
ＨＴ	＄０９	ＴＡＢ動作
ＬＦ	＄０Ａ	ラインフィールド
ＨＯＭＥ	＄０Ｂ	ホーム動作
ＥＡ	＄０Ｃ	画面クリア
ＣＲ	＄０Ｄ	復帰動作
ＳＦ	＄１１　アトリビート	フィールド定義
ＳＢＡ	＄１２　Ｘ　Ｙ	バッファアドレス指定
ＲＣ	＄１３　文字数　コード	指定数同一文字表示
→	＄１Ｃ	カーソル右移動
←	＄１Ｄ	カーソル左移動
↑	＄１Ｅ	カーソル上移動
↓	＄１Ｆ	カーソル下移動
Lock Keyboard	＄１Ｂ　＄２３	キー入力禁止
Unlock Keyboard	＄１Ｂ　＄２２	キー入力禁止解除
Erase Key buffer	＄１Ｂ　＄３９	キーバッファ消去
Set Buffer Mode	＄１Ｂ　＄６７	キー先行入力可
Set Unbuffer Mode	＄１Ｂ　＄６８	キー先行入力禁止

注） 例えば，ＢＡＳＩＣで

ＰＲＩＮＴ　ＣＨＲ＄（２７）＋〝g〟↲

とすれば，先行入力が可能になります．

カラーコード表

コード番号	G	R	B	色	補色
0	0	0	0	黒　(ブラック)	白
1	0	0	1	青　(ブルー)	黄
2	0	1	0	赤　(レッド)	水色
3	0	1	1	紫　(マゼンダ)	緑
4	1	0	0	緑　(グリーン)	紫
5	1	0	1	水色 (シアン)	赤
6	1	1	0	黄　(イエロー)	青
7	1	1	1	白　(ホワイト)	黒

ファンクション・コード表

コード	機能名	機能
0	PSET	指定色のドットを表示
1	PRESET	ドットを背景色にする
2	OR	出力色と画面色のORをとって表示
3	AND	出力色と画面色のANDをとって表示
4	XOR	出力色と画面色のXORをとって表示
5	NOT	ドットの補色をとる

アトリビュート文字

bit 7	6	5	4	3	2	1	0
F	M	×	P	R		CL	

ビット		
0～2	CL	文字色（カラーコード）
3	R	反転フラグ（1のとき文字を反転表示）
4	P	“0” 非保護 “1” 保護
6	M	変更フラグ　｝サブシステム内で設定されます．
7	F	フィールド先頭を示す　｝サブシステム内で設定されます．

2－2　各コマンド解説

ここでは，サブシステムのコマンドを，用途別に分けて解説します．

機　　能　各コマンドの機能を示します．

パラメータ　各コマンドを実行するに当って，設定すべきパラメータを示します．ただし，ここで示すのはサブシステムコマンドとしてのパラメータだけで，ＢＩＯＳ経由でサブシステムを呼び出すときの，ＢＩＯＳのコマンドやパラメータに関しては触れませんので，ＢＩＯＳの各リクエストの解説を御参照下さい．

復帰情報　各コマンドを実行した後に返される情報を示します．ただし，この復帰情報はサブシステムのコマンド実行が完全に終了しないと得られないので，ＢＩＯＳのＳＵＢＯＵＴを使用した場合には，エラー番号も含めて情報は返されません．

解　　説　各コマンドについて注意事項，詳しい解説，予備知識等を示します．

サンプル　いくつかのコマンドについてサンプルプログラムを付けています．サンプルプログラムはすべてＢＩＯＳを経由してサブシステムコマンドを実行する形をとっています．

ＳＵＢ：ＩＮＩＴ　ーコンソール初期化ー

機　能　コンソール機能の設定と初期化を行う.

パラメータ

0	$00	（　）内はリセット時のパラメータ
1	$00	
2	$01	コマンドコード
3	BC	背景色　カラーコード　０～７（０）
4	NC	桁数　　８０又は４０（８０）
5	NL	行数　　２５又は２０（２５）
6	SL	スクロール開始行（０）
7	NS	スクロール行数　（２５）
8	FD	ファンクションキー表示　０：表示しない.（０）
9	ERS	初期設定後の画面消去　０：非消去（１）
A	GR	単色（グリーン）表示　０：単色表示しない.（０）

注）ＦＤ≠０のときは，最後の２行をスクロール範囲に指定してはならない.

復帰情報

0	E	エラーコード

エラーが発生した時は，リセット時と同じ設定となります.

サンプル　各パラメータをＣＲで区切って入力して下さい．指定の通りに初期化します.

```
PAGE   001 (          ,          )

01000                           *
01010                           * SUBSYSTEM INIT
01020                           *
01030                                   OPT     M,NOS,NOG,NOO,PAGE=255
01040               9BDB        MESSAG EQU     $9BDB    output from (X+1) till 0
01050               D08E        OUT     EQU     $D08E    one character output
01060               FBFA        BIOS    EQU     $FBFA    BIOS (extend-indirect)
01070               AC3D        A16OUT EQU     $AC3D    output A reg in HEX
01080               DB54        KEYIN   EQU     $DB54    keyinput with wait
01090                           *
01100   5000                            ORG     $5000
01110               5000        START   EQU     *
01120                           *
```

```
01130  5000 31    8D 00F1    INIT   LEAY   SUBCOM,PCR load subsys-com. addr.
01131                        *
01132                        * set back color
01133                        *
01140  5004 30    8D 00A9           LEAX   MES1-1,PCR output message
01150  5008 BD    9BDB              JSR    MESSAG
01160  500B 17    0084 5092         LBSR   GETNUM   get number
01170  500E C4    07                ANDB   #7       back color=0..7
01180  5010 E7    23                STB    3,Y      set back color
01181                        *
01182                        * set width
01183                        *
01190  5012 30    8D 00AA    WIDTH  LEAX   MES2-1,PCR output message
01200  5016 BD    9BDB              JSR    MESSAG
01210  5019 8D    77   5092         BSR    GETNUM   get number
01220  501B C1    50                CMPB   #80      x=80 or 40
01230  501D 27    04   5023         BEQ    SETX
01240  501F C1    28                CMPB   #40
01250  5021 26    EF   5012         BNE    WIDTH
01260  5023 E7    24         SETX   STB    4,Y      set width x
01270  5025 86    2C                LDA    #',      output ','
01280  5027 BD    D08E              JSR    OUT
01290  502A 8D    66   5092         BSR    GETNUM   get number
01300  502C C1    19                CMPB   #25      y=25 or 20
01310  502E 27    04   5034         BEQ    SETY
01320  5030 C1    14                CMPB   #20
01330  5032 26    DE   5012         BNE    WIDTH
01340  5034 E7    25         SETY   STB    5,Y      set width y
01341                        *
01342                        * set console parameter
01343                        *
01350  5036 30    8D 008F    CONS   LEAX   MES3-1,PCR output message
01360  503A BD    9BDB              JSR    MESSAG
01370  503D 8D    53   5092         BSR    GETNUM   get number
01380  503F E1    25                CMPB   5,Y
01390  5041 24    F3   5036         BHS    CONS     if start_line > width y
01400  5043 E7    26                STB    6,Y      set start_line
01405                        * set num. of scroll & PF flag & Green flag
01410  5045 86    07                LDA    #7       counter
01420  5047 34    02         C01    PSHS   A        save counter
01430  5049 86    2C                LDA    #',      output ','
01440  504B BD    D08E              JSR    OUT
01450  504E 8D    42   5092         BSR    GETNUM   get number
01460  5050 A6    E4                LDA    ,S
01470  5052 E7    A6                STB    A,Y      set parameter
01480  5054 35    02                PULS   A        counter come back
01490  5056 4C                      INCA
01500  5057 81    09                CMPA   #9       skip clear flag
01510  5059 26    01   505C         BNE    C02
01520  505B 4C                      INCA
01530  505C 81    0B         C02    CMPA   #11      green flag end ?
01540  505E 26    E7   5047         BNE    C01
01541                        *
01542                        * check scroll line
01543                        *
01550  5060 A6    26                LDA    6,Y      load scroll start
01560  5062 AB    27                ADDA   7,Y      ;if PF on
01570  5064 6D    28                TST    8,Y      ;   width y>=SL+NS+2
01580  5066 27    02   506A         BEQ    C03      ;if PF off
01590  5068 8B    02                ADDA   #2       ;   width y>=SL+NS
01600  506A A1    25         C03    CMPA   5,Y
01610  506C 22    C8   5036         BHI    CONS
01611                        *
01612                        * set clear flag
01613                        *
01620  506E 30    8D 0062           LEAX   MES4-1,PCR output message
01630  5072 BD    9BDB              JSR    MESSAG
01640  5075 8D    1B   5092         BSR    GETNUM   get number
01650  5077 E7    29                STB    9,Y      set clear flag
01660  5079 C6    01                LDB    #$01     command INIT
01670  507B E7    22                STB    2,Y      set command
01680  507D 30    8D 006E           LEAX   RCB,PCR bios rcb set
01690  5081 86    10                LDA    #$10
01700  5083 A7    84                STA    ,X
```

```
01710  5085 10AF 02                  STY     2,X
01715  5088 CC   0080                LDD     #128
01720  508B ED   04                  STD     4,X
01730  508D AD   9F FBFA             JSR     [BIOS]
01740  5091 39                       RTS
02000                         *
02001                         * subroutine:get number 0..255 in B reg
02002                         *
02010  5092 5F                GETNUM CLRB
02020  5093 4F                       CLRA
02030  5094 34   02           GN01   PSHS    A        save keyinput
02040  5096 86   0A                  LDA     #10      Breg=Breg*10
02050  5098 3D                       MUL
02060  5099 EB   E0                  ADDB    ,S+
02070  509B 34   04                  PSHS    B
02080  509D BD   DB54                JSR     KEYIN    keyinput
02090  50A0 35   04                  PULS    B
02100  50A2 81   30                  CMPA    #'0      test '0'..'9'
02110  50A4 25   0B    50B1          BCS     GN02
02120  50A6 81   39                  CMPA    #'9
02125  50A8 22   07    50B1          BHI     GN02
02126  50AA BD   D08E                JSR     OUT      echoback
02130  50AD 80   30                  SUBA    #'0      '0'->0
02140  50AF 20   E3    5094          BRA     GN01
02160  50B1 39                GN02   RTS              num.0-255 in Breg
02170                         *
02171                         * messages
02172                         *
02180            0D0A         CRLF   EQU     $0D0A
02190  50B2      0D0A         MES1   FDB     CRLF
02200  50B4      42                  FCC     'Back Color ='
02210  50C0      00                  FCB     0
02220  50C1      0D0A         MES2   FDB     CRLF
02230  50C3      57                  FCC     'WIDTH '
02240  50C9      00                  FCB     0
02250  50CA      0D0A         MES3   FDB     CRLF
02260  50CC      43                  FCC     'CONSOLE '
02270  50D4      00                  FCB     0
02280  50D5      0D0A         MES4   FDB     CRLF
02290  50D7      63                  FCC     'clear=1 or NOclear=0 ? '
02300  50EE      00                  FCB     0
02310                         *
02311                         * working area
02312                         *
02320  50EF      0006         RCB    RMB     6        for bios
02330  50F5      000B         SUBCOM RMB     11       for subsys. command
02340                         *
09999            5000                END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50FF
PROGRAM ENTRY ADDR=5000
```

```
Ready
HARDC 2

exec &H5000

Back Color =0
WIDTH 40,20
CONSOLE 0,20,0,0
clear=1 or NOclear=0 ? 0
```

P.30の Getnum 같이.

SUB：ERASE・ERASE２　―画面クリアー

機　能　画面と，それに付属するアトリビュートバッファをクリアします．ＥＲＡＳＥ２には文字色指定がありますがＥＲＡＳＥにはありません．

パラメータ

0	$00	
1	$00	
2	C	コマンドコード　ERASE　：$02／ERASE2：$0D
3	W	消去範囲コード
4	BC	背景色カラーコード　0～7
5	FC	文字色カラーコード　0～15（ERASE2のみ）

W：消去範囲コード　　0：全画面
1：スクロールモード画面
2：ページモード１画面
3：ページモード２画面

復帰情報

0	E	エラーコード

解　説　背景色カラーコードの指定は消去範囲コードが0（全画面消去）のときのみ有効となります．

消去後，カーソルの位置（バッファアドレス）は消去した画面の先頭となります．

サンプル

```
PAGE  001 (         ,         )

01000                          *
01010                          * SUBSYSTEM ERASE2
01020                          *
01030                                 OPT     M,NOS,NOG,NOO,PAGE=255
01040              9BDB        MESSAG EQU     $9BDB   output from (X+1) till 0
01050              9C16        MESSA2 EQU     $9C16   output Breg bytes from (X)
01060              D08E        OUT    EQU     $D08E   one character output
01070              FBFA        BIOS   EQU     $FBFA   BIOS (extend-indirect)
01080              AC3D        A16OUT EQU     $AC3D   output A reg in HEX
01090              DB54        KEYIN  EQU     $DB54   keyinput with wait
01100                          *
01110  5000                           ORG     $5000
```

```
01120             5000       START  EQU     *
01130                        *
01140                        * set parameters
01150                        *
01160  5000 31    8D 009F    ERASE2 LEAY    SUBCOM,PCR
01170                        *
01180                        * set clear area no.
01190                        *
01200  5004 30    8D 0063           LEAX    MES1-1,PCR output message
01210  5008 BD    9BDB              JSR     MESSAG
01220  500B 8D    3F    504C        BSR     GETNUM   get number
01230  500D C4    03                ANDB    #3       no.=0..3
01240  500F E7    23                STB     3,Y      set clear area no.
01250                        *
01260                        * set character color code & back color code
01270                        *
01280  5011 30    8D 006A           LEAX    MES2-1,PCR output message
01290  5015 BD    9BDB              JSR     MESSAG
01300  5018 8D    32    504C        BSR     GETNUM   get number
01310  501A C4    0F                ANDB    #15      chr. color 0..15
01320  501C E7    25                STB     5,Y      set chr. color
01330  501E 86    2C                LDA     #',      output ','
01340  5020 BD    D0BE              JSR     OUT
01350  5023 8D    27    504C        BSR     GETNUM   get number
01360  5025 C4    07                ANDB    #7       back color 0..7
01370  5027 E7    24                STB     4,Y      set back color
01380                        *
01390                        * set subsystem command
01400                        *
01410  5029 CC    100D              LDD     #$100D   subsys. command ERASE2
01420  502C E7    22                STB     2,Y      set subsys. command
01430  502E 30    8D 006B           LEAX    RCB,PCR
01440  5032 A7    84                STA     ,X       set rcb request no. (SUBOUT)
01450  5034 10AF  02                STY     2,X
01460  5037 CC    0080              LDD     #128
01470  503A ED    04                STD     4,X
01480  503C AD    9F FBFA           JSR     [BIOS]   command exec.
01490  5040 30    8D 0046           LEAX    MES3,PCR output message
01500  5044 E6    8D 0041           LDB     MES3L,PCR
01510  5048 BD    9C16              JSR     MESSA2
01520  504B 39                      RTS
01530                        *
01540                        * subroutine:get number 0..255 in Breg
01550                        *
01560  504C 5F               GETNUM CLRB
01570  504D 4F                      CLRA
01580  504E 34    02         GN01   PSHS    A        save keyinput
01590  5050 86    0A                LDA     #10      Breg=Breg*10
01600  5052 3D                      MUL
01610  5053 EB    E0                ADDB    ,S+      Breg=Breg+keyinput
01620  5055 34    04                PSHS    B
01630  5057 BD    DB54              JSR     KEYIN
01640  505A 35    04                PULS    B
01650  505C 81    30                CMPA    #'0      test '0'..'9'
01660  505E 25    0B    506B        BCS     GN02
01670  5060 81    39                CMPA    #'9
01680  5062 22    07    506B        BHI     GN02
01690  5064 BD    D08E              JSR     OUT
01700  5067 80    30                SUBA    #'0      '0'->0
01710  5069 20    E3    504E        BRA     GN01
01720  506B 39               GN02   RTS              num.0-255 in Breg
01730                        *
01740                        * messages
01750                        *
01760             0D0A       CRLF   EQU     $0D0A
01770  506C       0D0A       MES1   FDB     CRLF
01780  506E       43                FCC     'Clear Area (0-3)='
01790  507F       00                FCB     0
01800  5080       0D0A       MES2   FDB     CRLF
01810  5082       43                FCC     'Color
01820  5088       00                FCB     0
01830  5089       13         MES3L  FCB     .MES3-MES3
01840  508A       45         MES3   FCC     'End of execution !!'
01850             509D       .MES3  EQU     *
```

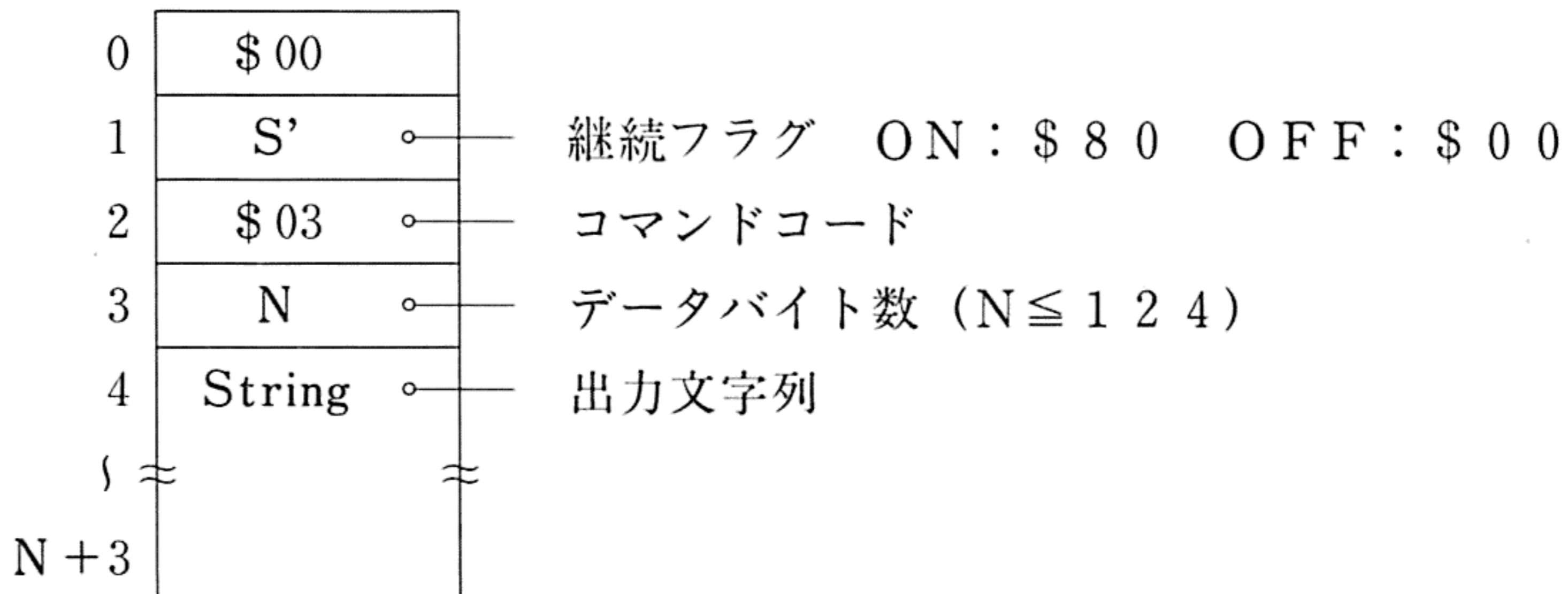
```
01860                          *
01870                          * working area
01880                          *
01890  509D       0006         RCB     RMB     6
01900  50A3       0006         SUBCOM RMB      6
01910                          *
01920             5000                 END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50A8
PROGRAM ENTRY ADDR=5000
```

```
exec &H5000

Clear Area (0-3)=0
Color 7,5End of execution !!
Ready
```

ＳＵＢ：ＰＵＴ　―文字列出力―

機　能　文字列をコンソール画面に表示します．文字列の中にオーダーが含まれている場合は，該当するオーダ動作を行います．

パラメータ

0	$00	
1	S'	継続フラグ　ON：$80　OFF：$00
2	$03	コマンドコード
3	N	データバイト数（N≦124）
4	String	出力文字列
∫		
N+3		

出力したい文字列が124バイトを越える場合は，継続フラグをONにしてコマンドを送り，その後，CONTコマンドを用いて残りのデータを送ります．さらに続く場合も同様にし，最後の転送のときに，継続フラグをOFFにしてコマンドを送ります．

復帰情報

0	E	エラーコード

サンプル

```
PAGE   001 (         ,         )

01000                          *
01010                          * SUBSYSTEM PUT
01020                          *
01030                                 OPT     M,NOS,G,NOO,PAGE=255
01040               FBFA       BIOS   EQU     $FBFA    BIOS (extend-indirect)
01050                          *
01060   5000                          ORG     $5000
01070               5000       START  EQU     *
01080                          *
01090                          * set characters
01100                          *
01110   5000 31     8D 0051           LEAY    SUBCOM,PCR load subcom. addr.
01120   5004 86     03                LDA     #$03     subsys-command PUT
01130   5006 A7     22                STA     2,Y      set command
01140   5008 33     24                LEAU    4,Y      load string addr.
01150   500A C6     05                LDB     #5       counter1
01160   500C 34     04                PSHS    B        save counter1
01170   500E 5F                       CLRB             clear counter2
01180   500F 86     20         LOOP01 LDA     #'       space !
01190   5011 A7     C0         LOOP02 STA     ,U+      set character
```

```
01200  5013 5C                 INCB           inc counter
01210  5014 C1   7C            CMPB   #128-4  buffer full ?
01220  5016 26   06   501E     BNE    NOOUT   no.
01230  5018 34   02            PSHS   A
01240  501A 8D   0F   502B     BSR    CONT    yes,output string
01250  501C 35   02            PULS   A
01260  501E 4C           NOOUT INCA           inc character
01270  501F 81   FF            CMPA   #255    chr. ' '..$FE end ?
01280  5021 26   EE   5011     BNE    LOOP02  no.
01290  5023 6A   E4            DEC    ,S      yes,dec counter1
01300  5025 26   E8   500F     BNE    LOOP01
01310  5027 8D   05   502E     BSR    NOCONT  output string
01320  5029 35   84            PULS   B,PC    end of execution
01330                    *
01340                    * output string
01350                    *
01360  502B 86   80      CONT   LDA   #$80    set cont flag
01370  502D      81             FCB   $81     = CMPA # (skip next op-code)
01380  502E 4F           NOCONT CLRA          clear cont flag
01390  502F A7   21             STA   1,Y
01400  5031 E7   23             STB   3,Y     set string length
01410  5033 30   8D 0018        LEAX  RCB,PCR load rcb address
01420  5037 86   10             LDA   #$10    bios request SUBOUT
01430  5039 A7   84             STA   ,X      set request no.
01440  503B 10AF 02             STY   2,X     set subsys-com. addr.
01450  503E CC   0080           LDD   #128    set command length
01460  5041 ED   04             STD   4,X
01470  5043 AD   9F FBFA        JSR   [BIOS]  command execution
01480  5047 86   64             LDA   #$64    subsys-command CONT
01490  5049 A7   22             STA   2,Y     set command
01500  504B 5F                  CLRB          clear counter2
01510  504C 33   24             LEAU  4,Y     load string address
01520  504E 39                  RTS
01530                    *
01540                    * working area
01550                    *
01560  504F      0006    RCB    RMB   6
01570  5055      0080    SUBCOM RMB   128
01580                    *
01590            5000           END   START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50D4
PROGRAM ENTRY ADDR=5000
```

```
exec &H5000
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[¥]^_`abcdefghijklmno
pqrstuvwxyz{|}~▁▂▃▄▅▆▇█▏▎▍▌▋▊▉┼┴┬┤├▔─│▕┌┐└┘╭╮╰╯ 。「」、・ヲァィゥェォャュョッーアイウエオカキクケコサシスセソタ
チツテトナニヌネノハヒフヘホマミムメモヤユヨラリルレロワン゛゜＝┣┫◢◣◥◤♠♥♦♣●○╱╲╳円年月日時分秒千市区町村人▒ !"#$%&'()*+,-./01
23456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[¥]^_`abcdefghijklmnopqrstuvwxyz{|}~▁▂
▃▄▅▆▇█▏▎▍▌▋▊▉┼┴┬┤├▔─│▕┌┐└┘╭╮╰╯ 。「」、・ヲァィゥェォャュョッーアイウエオカキクケコサシスセソタチツテトナニヌネノハヒフヘホマミムメ
モヤユヨラリルレロワン゛゜＝┣┫◢◣◥◤♠♥♦♣●○╱╲╳円年月日時分秒千市区町村人▒ !"#$%&'()*+,-./0123456789:;<=>?@ABC
DEFGHIJKLMNOPQRSTUVWXYZ[¥]^_`abcdefghijklmnopqrstuvwxyz{|}~▁▂▃▄▅▆▇█▏▎▍▌▋▊▉┼┴┬┤├▔
─│▕┌┐└┘╭╮╰╯ 。「」、・ヲァィゥェォャュョッーアイウエオカキクケコサシスセソタチツテトナニヌネノハヒフヘホマミムメモヤユヨラリルレロワン゛゜＝┣┫◢
◣◥◤♠♥♦♣●○╱╲╳円年月日時分秒千市区町村人▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[¥]^_`abcdefghijklmnopqrstuvwxyz{|}~▁▂▃▄▅▆▇█▏▎▍▌▋▊▉┼┴┬┤├▔─│▕┌┐└┘╭╮╰╯ 。「」、・ヲ
ァィゥェォャュョッーアイウエオカキクケコサシスセソタチツテトナニヌネノハヒフヘホマミムメモヤユヨラリルレロワン゛゜＝┣┫◢◣◥◤♠♥♦♣●○╱╲╳円年月日時分
秒千市区町村人▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[¥]^_`abcdefg
hijklmnopqrstuvwxyz{|}~▁▂▃▄▅▆▇█▏▎▍▌▋▊▉┼┴┬┤├▔─│▕┌┐└┘╭╮╰╯ 。「」、・ヲァィゥェォャュョッーアイウエオカキク
ケコサシスセソタチツテトナニヌネノハヒフヘホマミムメモヤユヨラリルレロワン゛゜＝┣┫◢◣◥◤♠♥♦♣●○╱╲╳円年月日時分秒千市区町村人▒
Ready
```

ＳＵＢ：ＧＥＴ　―文字列入力―

機　能　文字列を出力後，入力待ちとなり入力された文字列を受け取ります．

パラメータ

0	$00	
1	S'	継続フラグ　継続：＄８０，終了：＄００
2	$04	コマンドコード
3	N	出力文字数（N≦１２４）
4	String	出力文字列
～		
N＋3		

出力文字が１２４バイトを越える場合にはＰＵＴと同じ方法でデータを継続して下さい．

復帰情報　文字列出力後ユーザに変更されたフィールドのうち最上段のフィールドを返します．

0	E	エラーコード
1	S'	継続フラグ（bit 7）（＊）
2		
3	K	終了キーコード　＄００：Return　＄０１:∧C or X　＄０２：ＣＬＳ
4	N	転送バイト数（ＳＢＡ，Ｘ，Ｙをふくむ）
5	SBA	オーダＳＢＡ＝＄１２
6	X	変更フィールドのＸ座標
7	Y	変更フィールドのＹ座標
8	String	変更フィールドの文字列
～		
N＋4		

＊：継続フラグ（bit 7）がＯＮのときは，文字列がまだ続くので，ＣＯＮＴコマンドを使用して残りを受けとる必要があります．

解　説　転送される文字列の中には，オーダコードは含まれません．このコマンドでは変更されたフィールドのうち最上段のものを転送するので，ユーザーは，残りの変更されたフィールドをＧＥＴＣコマンドを用いて受け取る必要があ

ります．

受け取った場合に，N＝0ならばすべてのフィールドを転送し終ったことを示し，N＝3ならば変更されたフィールドが空文字列であることを示します．

ＳＵＢ：ＧＥＴＣ　一文字列継続入力一

機　能　ＧＥＴの後で，残りの変更フィールドを受け取ります．

パラメータ

0	$00	
1	$00	
2	$05	コマンドコード

復帰情報　まだユーザーに返されていないフィールドのうちで最上段のフィールドを返します．

フォーマットはＧＥＴと同様です．

解　説　このコマンドはＧＥＴコマンド実行後に使用し，残りのフィールドを受け取ります．ユーザは，変更されたフィールドをすべて受け取るまで（N＝0となるまで），ＧＥＴＣコマンドをくり返す必要があります．

SUB：GCBLK1　―枠内文字コード読み取り―

機　　能　キャラクタ座標で指定した枠内の文字コードを読み取ります．

パラメータ

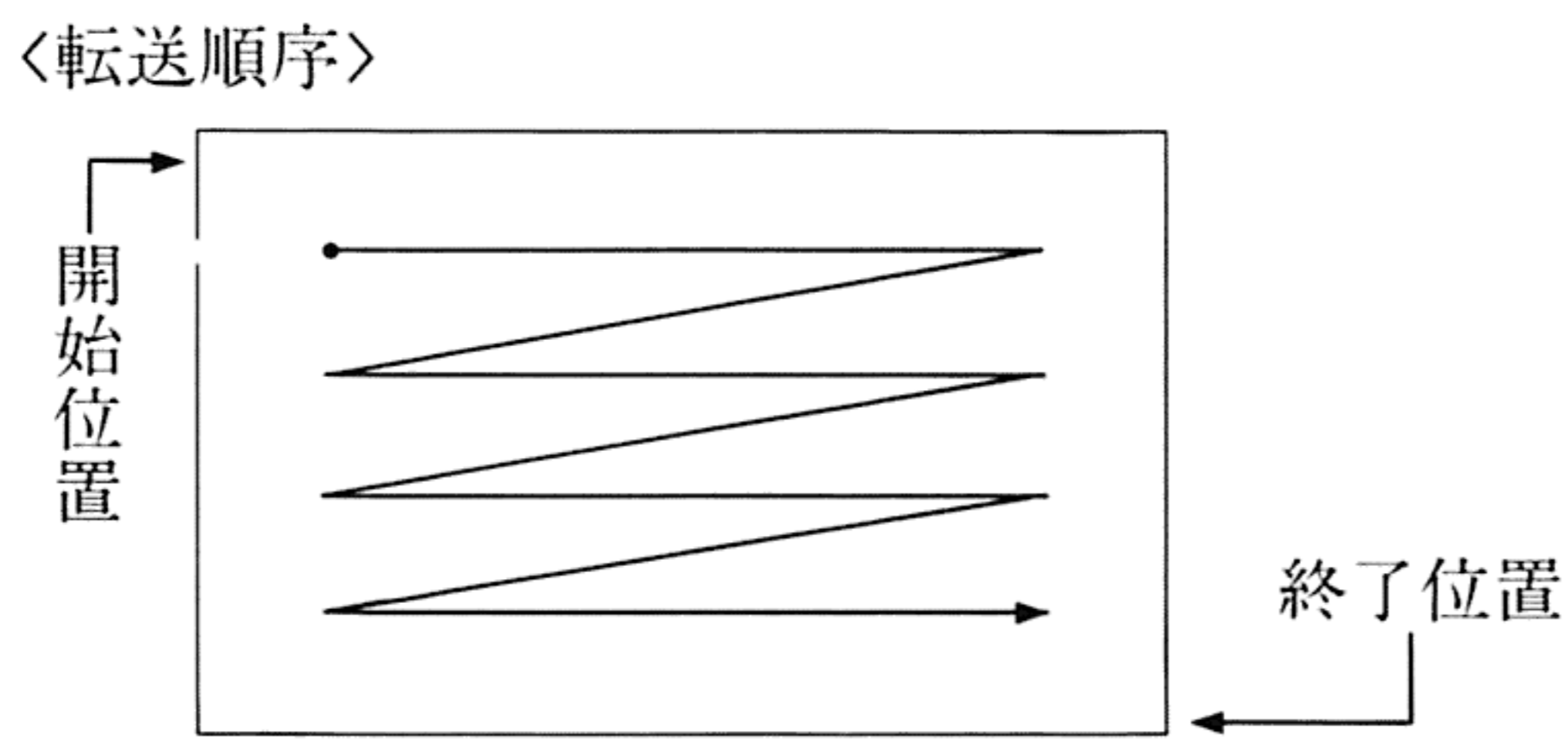
0	$00	
1	$00	
2	$06	コマンドコード
3	X1	開始位置のキャラクタ座標　X（0～79）
4	Y1	Y（0～24）
5	X2	終了位置のキャラクタ座標　X（0～79）
6	Y2	Y（0～24）

復帰情報

0	E	エラーコード
1	S'	継続フラグ（bit 7）
2		
3	B	転送バイト数
4	C1	転送文字列
∫	∫	
B＋3	Cm	

継続フラグがONのときはCONTコマンドで残りのデータを要求する必要があります．

解　　説　このコマンドは文字コードのみを転送し，その文字の色等の情報は転送されません．

〈転送順序〉

開始位置

終了位置

SUB：PCBLK1　―枠内文字コード表示―

機　能　GCBLK1と逆に枠内に文字を表示します.

パラメータ

0	$00	
1	S'	継続フラグ（bit 7）
2	$07	コマンドコード
3	X1	開始位置のキャラクタ座標　X（0～79）
4	Y1	Y（0～24）
5	X2	終了位置のキャラクタ座標　X（0～79）
6	Y2	Y（0～24）
7	at	アトリビュート（色指定）文字
8	B	転送バイト数
9	C_1	転送文字列
～	～	
B+8	Cm	

分割して転送する場合は継続フラグを1にしてコマンドを送り，残りのデータをCONTコマンドで転送します.

復帰情報

0	E	エラーコード

解　説　文字の転送順序はGCBLK1と同様です.

SUB:GCBLK2 —枠内文字コード及び属性読み取り—

機　能 キャラクタ座標で指定した枠内の文字コードを属性とともに読み取ります.

パラメータ

0	$00	
1	$00	
2	$08	コマンドコード
3	X1	開始位置のキャラクタ座標　X（0～79）
4	Y1	Y（0～24）
5	X2	終了位置のキャラクタ座標　X（0～79）
6	Y2	Y（0～24）

復帰情報

0	E	エラーコード
1	S'	継続フラグ（bit 7）
2		
3	B	転送バイト数（1～124）
4	at_1	アトリビュート文字
5	C_1	文字コード
6	at_2	
7	C_2	
～	～	
B+2	at_m	
B+3	C_m	

継続フラグがONのときは，CONTコマンドを使用して残りのデータを要求する必要があります.

解　説 アトリビュート文字及び文字コードの読み取り順序は,GCBLK1と同様です.

SUB：PCBLK 2　一枠内文字コード及び属性表示一

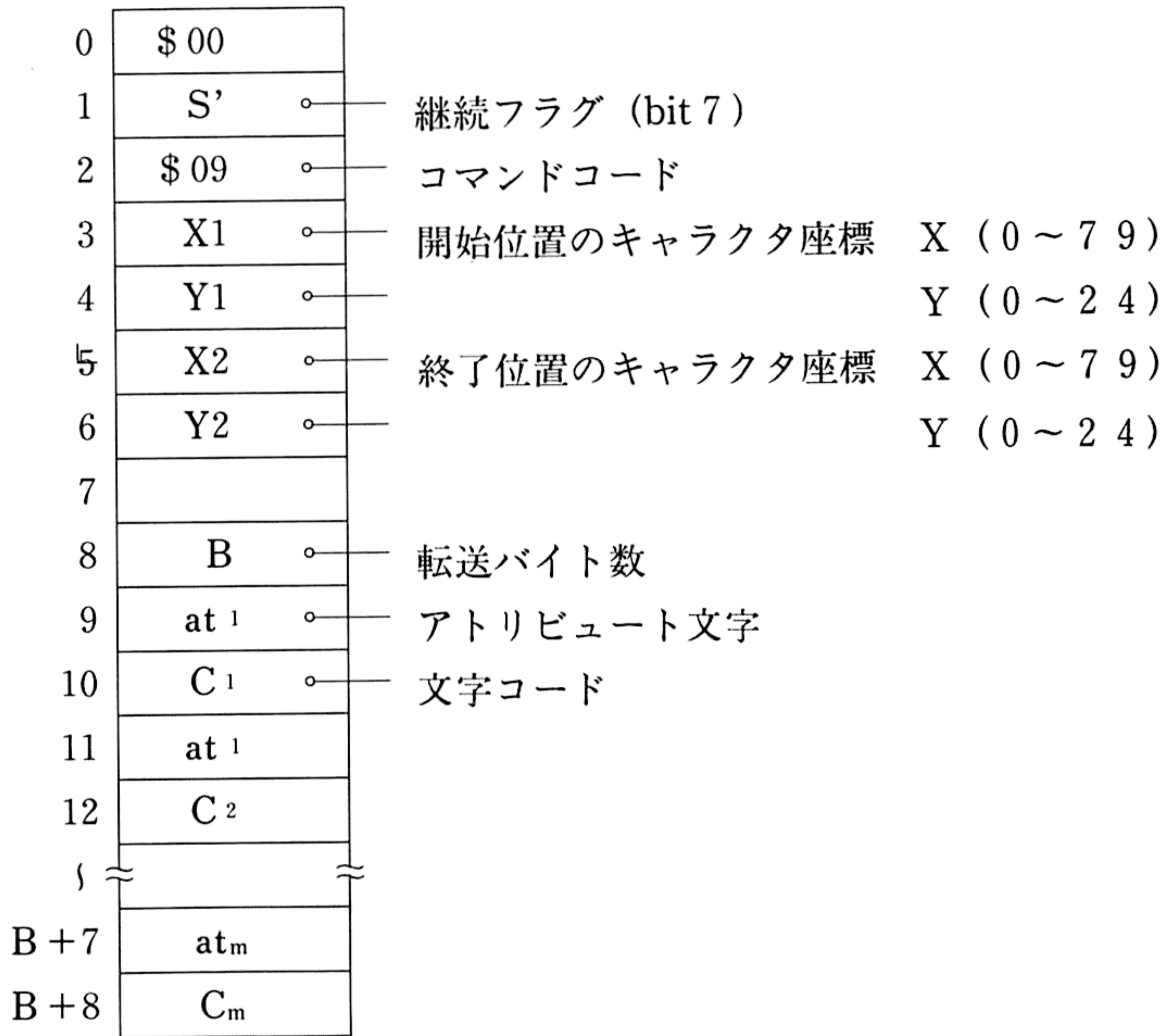
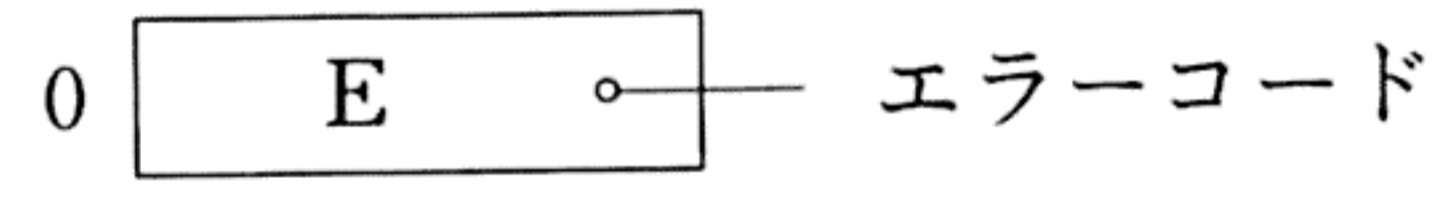
機　能　GCBLK 2と逆に枠内に文字を表示します．この際，属性の指定も行います．

パラメータ

0	$ 00	
1	S'	継続フラグ（bit 7）
2	$ 09	コマンドコード
3	X1	開始位置のキャラクタ座標　X（0～7 9）
4	Y1	Y（0～2 4）
5	X2	終了位置のキャラクタ座標　X（0～7 9）
6	Y2	Y（0～2 4）
7		
8	B	転送バイト数
9	at 1	アトリビュート文字
10	C 1	文字コード
11	at 1	
12	C 2	
⁓	⁓	
B＋7	at_m	
B＋8	C_m	

分割して転送する場合は継続フラグを１にしてコマンドを送り，残りのデータをＣＯＮＴコマンドで転送します．

復帰情報

0	E	エラーコード

解　説　アトリビュート文字及び文字コードの表示順序はＧＣＢＬＫ１と同様です．

サンプル　ＧＣＢＬＫ２及びＰＣＢＬＫ２を使用した例です．

```
100 '
110 ' GCBLK2 & PCBLK2 sample
120 '
```

```
130 RANDOMIZE TIME
140 WIDTH 80,25:CLS
150 LOCATE 25,15
160 PRINT "*** sample animation ***"
170 LOCATE 0,0
180 GOSUB 390:PRINT" ╭─────╮"
190 GOSUB 390:PRINT" |●○○●|"
200 GOSUB 390:PRINT" ═══════"
210 GOSUB 390:PRINT" ／／==＼＼"
220 EXEC &H5000
230 LOCATE 0,0
240 PRINT
250 GOSUB 390:PRINT" ＼      "
260 GOSUB 390:PRINT"×══════>"
270 GOSUB 390:PRINT" ／      "
280 EXEC &H5000
290 GOSUB 390:LOCATE 80-7,0:PRINT " ┬┘┤┤├└
300 GOSUB 390:LOCATE 80-7,1:PRINT " ┐┐／／├┤
310 GOSUB 390:LOCATE 80-7,2:PRINT " ×○○○○×
320 GOSUB 390:LOCATE 80-7,3:PRINT " !BOMB!
330 GOSUB 390:LOCATE 80-7,4:PRINT " ×○○○○×
340 GOSUB 390:LOCATE 80-7,5:PRINT " ／├┌┐┬└
350 GOSUB 390:LOCATE 80-7,6:PRINT " ├└┘┤├┼
360 COLOR7:LOCATE 0,2:PRINT SPC(80-7);
370 LOCATE 0,20
380 END
390 COLOR INT(RND*7+1)
400 RETURN
```

```
PAGE  001 (        ,        )

01000                              *
01010                              * SUBSYSTEM GCBLK2 & PCBLK2
01020                              *
01030                                      OPT     M,NOS,NOG,NOO,PAGE=255
01040                 9BDB         MESSAG EQU     $9BDB    output from (X+1) till 0
01050                 FBFA         BIOS    EQU     $FBFA    bios (extend-indirect)
01060                 AC3D         A16OUT EQU     $AC3D    output A reg in HEX
01070                              *
01080  5000                                ORG     $5000
01090                 5000         START   EQU     *
01100                              *
01110  5000 20        02    5004           BRA     GET
01120                              *
01130                              * constant
01140                              *
01150  5002           06           LENX    FCB     6        move (lenx+1)*(leny+1) chr
01160  5003           06           LENY    FCB     6
01170                              *
01180                              * get character block
01190                              *
01200  5004 31        8D 00D3      GET     LEAY    SUBCOM,PCR
01210  5008 33        8D 014F              LEAU    DATA,PCR load data buffer address
01220  500C CC        0000                 LDD     #0       chr.adr. initialize
01230  500F ED        8D 00BC              STD     XADR,PCR
01240  5013 8D        58    506D           BSR     SETADR   setchradr.
01250  5015 CC        1108                 LDD     #$1108   subsys-command CGBLK2
01260  5018 8D        72    508C G01       BSR     EXEC     exec subsys-command
01270  501A 30        24                   LEAX    4,Y      skip 0-3
01280  501C E6        23                   LDB     3,Y      load data length
01290  501E A6        80           G02     LDA     ,X+      trans. to data buffer
01300  5020 A7        C0                   STA     ,U+
01310  5022 5A                             DECB
01320  5023 26        F9    501E           BNE     G02
01330  5025 6D        21                   TST     1,Y      trans. end ?
01340  5027 2A        05    502E           BPL     PUT      (CONT flag off)
01350  5029 CC        1164                 LDD     #$1164   subsys-command CONT
01360  502C 20        EA    5018           BRA     G01
01370                              *
01380                              * put character block
01390                              *
```

```
01400  502E EF   8D 009F  PUT    STU    DEND,PCR set data buffer end addr.
01410  5032 8D   39   506D P00   BSR    SETADR  set_chr_addr
01420  5034 33   8D 0123         LEAU   DATA,PCR get data buffer addr.
01430  5038 CC   0909            LDD    #$0909  subsys-com. CPBLK2 & skip 0-8
01440  503B A7   22              STA    2,Y     set subsys-command
01450  503D 34   04       P01    PSHS   B
01460  503F A6   C0       P02    LDA    ,U+     trans. to subcom
01470  5041 A7   A5              STA    B,Y
01480  5043 5C                   INCB
01490  5044 11A3 8D 0088         CMPU   DEND,PCR data end ?
01500  5049 27   06   5051       BEQ    P03
01510  504B 5D                   TSTB           subcom full (over 128 bytes)?
01520  504C 2A   F1   503F       BPL    P02
01530  504E 86   80              LDA    #$80    set CONT flag
01540  5050      81              FCB    $81     = CMPA #
01550  5051 4F            P03    CLRA           clear CONT flag
01560  5052 A7   21              STA    1,Y
01570  5054 E0   E4              SUBB   ,S      get num. of transfer
01580  5056 35   02              PULS   A
01590  5058 4A                   DECA
01600  5059 E7   A6              STB    A,Y     set num. of transfer
01610  505B 86   10              LDA    #$10    bios SUBOUT
01620  505D 8D   2F   508E       BSR    EX01    execute subsys-com.
01630  505F 11A3 8D 006D         CMPU   DEND,PCR data end ?
01640  5064 27   CC   5032       BEQ    P00
01650  5066 CC   6404            LDD    #$6404  set subsys-com. CONT & skip 0-
01660  5069 A7   22              STA    2,Y
01670  506B 20   D0   503D       BRA    P01
01680                     *
01690                     * subroutine:set character address
01700                     *
01710  506D EC   8D 005E  SETADR LDD    XADR,PCR set character adr.
01720  5071 ED   23              STD    3,Y     set X1,Y1
01730  5073 E3   8C 8C           ADDD   LENX,PCR
01740  5076 ED   25              STD    5,Y     set X2,Y2
01750  5078 EC   8D 0053         LDD    XADR,PCR
01760  507C 4C                   INCA            X1=X1+1
01770  507D ED   8D 004E         STD    XADR,PCR
01780  5081 AB   8D FF7D         ADDA   LENX,PCR X2 > 80 ?
01790  5085 80   50              SUBA   #80
01800  5087 26   02   508B       BNE    S01
01810  5089 35   10              PULS   X
01820  508B 39            S01    RTS
01830                     *
01840                     * subroutine:subsys-command execution
01850                     *
01860  508C E7   22       EXEC   STB    2,Y     set subsys-command
01870  508E 30   8D 0041  EX01   LEAX   RCB,PCR set bios parameter
01880  5092 A7   84              STA    ,X
01890  5094 10AF 02              STY    2,X
01900  5097 CC   0080            LDD    #128
01910  509A ED   04              STD    4,X
01920  509C ED   06              STD    6,X
01930  509E AD   9F FBFA         JSR    [BIOS]
01940  50A2 25   01   50A5       BCS    ERROR
01950  50A4 39                   RTS
01960                     *
01970                     * error message output
01980                     *
01990  50A5 A6   01       ERROR  LDA    1,X
02000  50A7 34   02              PSHS   A
02010  50A9 30   8D 0009         LEAX   ERRMES-1,PCR
02020  50AD BD   9BDB            JSR    MESSAG
02030  50B0 35   02              PULS   A
02040  50B2 BD   AC3D            JSR    A16OUT
02050  50B5 35   90              PULS   X,PC
02060                     *
02070                     * working area
02080                     *
02090  50B7      0D0A     ERRMES FDB    $0D0A
02100  50B9      53              FCC    'SUBSYSTEM ERROR no.=$'
02110  50CE      00              FCB    0
02120  50CF      0000     XADR   FDB    0
02130  50D1      0000     DEND   FDB    0
```

```
02140  50D3        0008        RCB     RMB     8
02150  50DB        0080        SUBCOM  RMB     128
02160              515B        DATA    EQU     *
02170                          *
02180              5000                END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=515A
PROGRAM ENTRY ADDR=5000
```

SUB：GBADR　―バッファアドレス読み取り―

機　能　現在のバッファアドレスを読み取ります．

パラメータ

0	$00	
1	$00	
2	$0A	コマンドコード

復帰情報

0	$00			
1	／			
2	／			
3	X	バッファアドレス	X座標	(0～79)
4	Y		Y座標	(0～24)

解　説　このコマンドは常に正常に終了しますのでエラーコードは常に0となります．

注)「ユーザーズマニュアルシステム仕様」の復帰情報の項の相対値4，5は，3，4の誤りです．

サンプル　バッファアドレスを画面左上に表示します．

```
PAGE  001 (         ,          )

01000                         *
01010                         * SUBSYSTEM GBADR
01020                         *
01030                                  OPT    M,NOS,NOG,NOO,PAGE=255
01040               FBFA      BIOS     EQU    $FBFA    ;bios (extend-indirect)
01050               D678      ABORTT   EQU    $D678    abort test
01060                         *
01070     5000                         ORG    $5000
01080               5000      START    EQU    *
01090                         *
01100     5000 31   8D 00B3   GBADR    LEAY   SUBCOM,PCR Y reg=subsys-com. base addr
01110     5004 20   1B   5021          BRA    G02
01120                         *
01130                         * keyinput & echoback
01140                         *
01150     5006 86   01        G01      LDA    #%00000001 keyinput with wait
01160     5008 A7   23                 STA    3,Y
01170     500A CC   1129               LDD    #$1129   subsys-command INKEY
01180     500D 8D   68   5077          BSR    EXEC
01190     500F 6F   21                 CLR    1,Y      echo-back !
01200     5011 A6   23                 LDA    3,Y      get chr
01210     5013 A7   24                 STA    4,Y      set chr
01220     5015 86   01                 LDA    #1       set chr length =1
01230     5017 A7   23                 STA    3,Y
01240     5019 CC   1003               LDD    #$1003   subsys-command PUT
01250     501C 8D   59   5077          BSR    EXEC
01260     501E BD   D678               JSR    ABORTT   break key test
```

```
01270                           *
01280                           * get buffer address
01290                           *
01300  5021 CC    110A          GO2    LDD    #$110A   subsys-command GBADR
01310  5024 8D    51    5077           BSR    EXEC
01320                           *
01330                           * display buffer address
01340                           *
01350  5026 EC    23                   LDD    3,Y        get buffer address
01360  5028 ED    8D 0081              STD    BADR,PCR
01370  502C 8D    3A    5068           BSR    HIGH       addr X to string
01380  502E A7    8D 0071              STA    XH,PCR
01390  5032 A6    8D 0077              LDA    BADR,PCR
01400  5036 8D    34    506C           BSR    LOW
01410  5038 A7    8D 0068              STA    XH+1,PCR
01420  503C A6    8D 006E              LDA    BADR+1,PCR addr Y to string
01430  5040 8D    26    5068           BSR    HIGH
01440  5042 A7    8D 0063              STA    YH,PCR
01450  5046 A6    8D 0064              LDA    BADR+1,PCR
01460  504A 8D    20    506C           BSR    LOW
01470  504C A7    8D 005A              STA    YH+1,PCR
01480  5050 30    8D 003A              LEAX   MES,PCR message output
01490  5054 33    24                   LEAU   4,Y
01500  5056 C6    21                   LDB    #.MES-MES set message length
01510  5058 E7    23                   STB    3,Y
01520  505A A6    80            GO3    LDA    ,X+        transfer message to subcom.
01530  505C A7    C0                   STA    ,U+
01540  505E 5A                         DECB
01550  505F 26    F9    505A           BNE    GO3
01560  5061 CC    1003                 LDD    #$1003   subsys-command PUT
01570  5064 8D    11    5077           BSR    EXEC
01580  5066 20    9E    5006           BRA    GO1
01590                           *
01600                           * subroutine:convert hexadecimal into string
01610                           *
01620  5068 46                   HIGH   RORA              MSN to String
01630  5069 46                          RORA
01640  506A 46                          RORA
01650  506B 46                          RORA
01660  506C 84    0F            LOW    ANDA   #$0F       LSN to String
01670  506E 81    0A                   CMPA   #$0A       ;if 0-9 then +$30 ('0')
01680  5070 25    02    5074           BLO    NO_9       ;if A-F then +$41('A')
01690  5072 8B    07                   ADDA   #'A-'9-1 add 7 (=$41-$30)
01700  5074 8B    30            NO_9   ADDA   #'0
01710  5076 39                         RTS
01720                           *
01730                           * subroutine:subsystem-command execution
01740                           *
01750  5077 30    8D 0034       EXEC   LEAX   RCB,PCR load rcb address
01760  507B A7    84                   STA    ,X         set bios command
01770  507D E7    22                   STB    2,Y        set subsys-command
01780  507F 10AF  02                   STY    2,X        set buffer address
01790  5082 CC    0025                 LDD    #.MES-MES+4 set num. of transfer bytes
01800  5085 ED    04                   STD    4,X
01810  5087 ED    06                   STD    6,X
01820  5089 AD    9F FBFA              JSR    [BIOS]  call bios !
01830  508D 39                         RTS
01840                           *
01850                           * messages
01860                           *
01870  508E       12            MES    FCB    $12,$0,$0 locate 0,0
01880  5091       42                   FCC    'Buffer Address X=$'
01890  50A3       00            XH     FCB    0,0        addr X
01900  50A5       20                   FCC    ' Y=$'
01910  50A9       00            YH     FCB    0,0        addr Y
01920  50AB       20                   FCC    ' '
01930  50AC       12                   FCB    $12        locate X,Y
01940  50AD       00            BADR   FCB    0,0
01950             50AF          .MES   EQU    *
01960                           *
01970                           * working area
01980                           *
01990  50AF       0008          RCB    RMB    8
02000  50B7       0025          SUBCOM RMB    .MES-MES+4
```

```
02010                             *
02020              5000                 END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50DB
PROGRAM ENTRY ADDR=5000


Buffer Address X=$16 Y=$04


exec &H5000
buffer address ------>● Here

Abort
Ready
HARDC
```

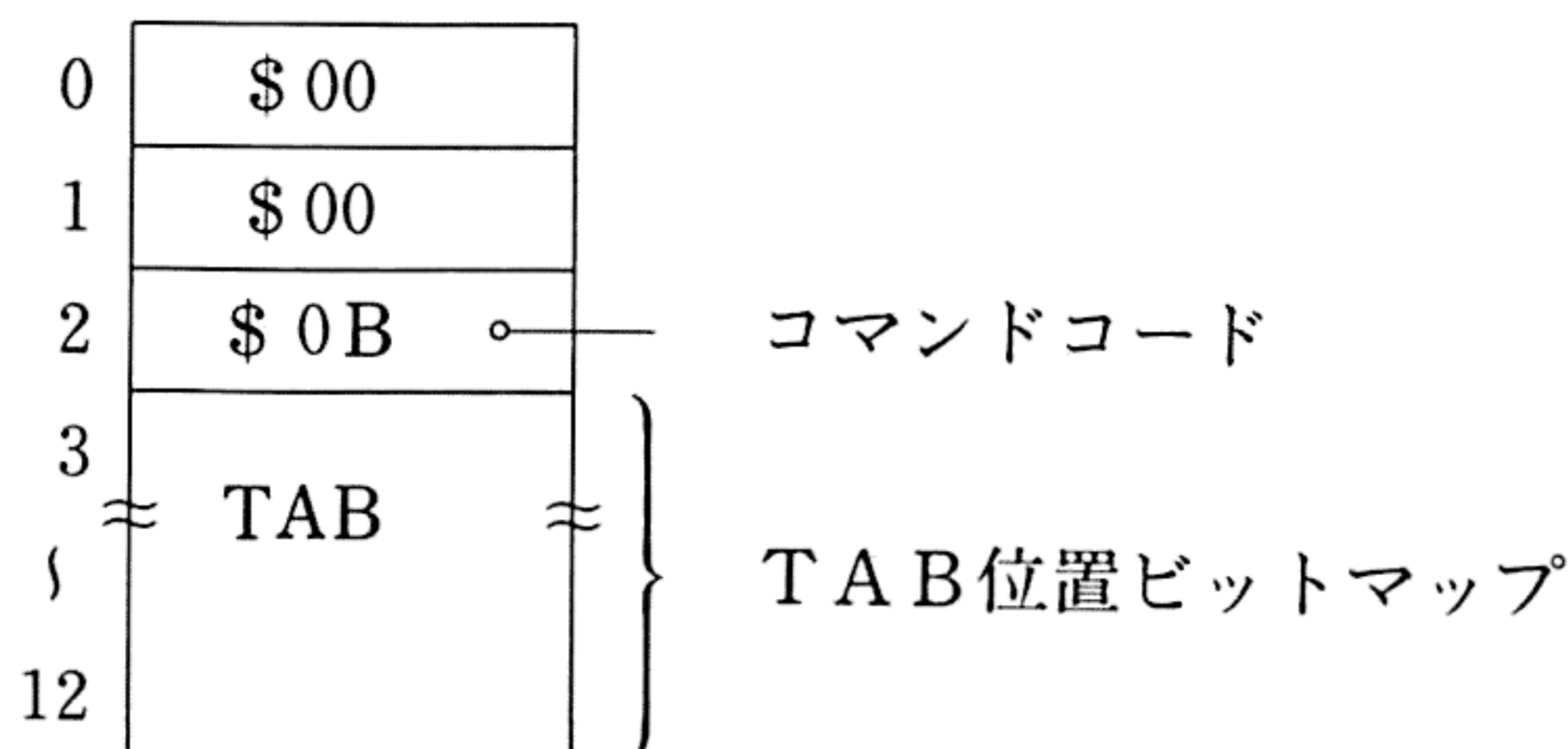
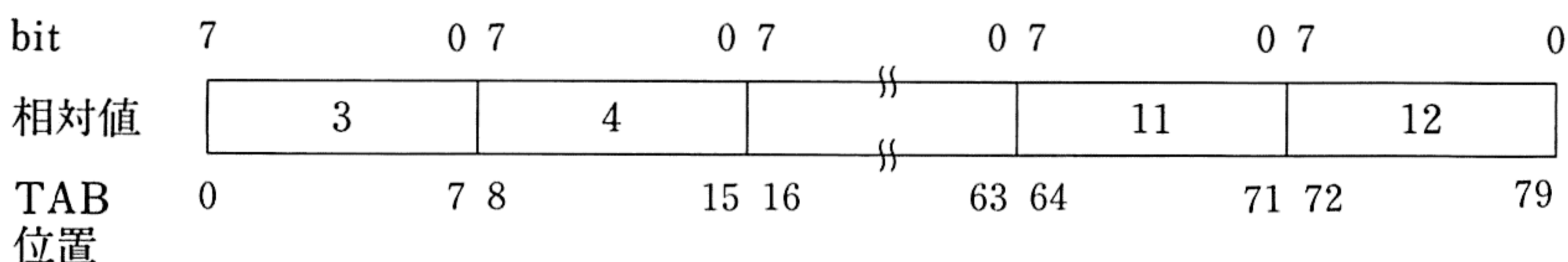
ＳＵＢ：ＴＡＢＳＥＴ　―ＴＡＢ位置設定―

機　能　タブキーを押した時のカーソル停止位置を設定します．

パラメータ

0	$ 00
1	$ 00
2	$ 0B ―― コマンドコード
3～12	TAB ―― ＴＡＢ位置ビットマップ

TAB位置ビットマップ

bit	7　　　0	7　　　0	7　　　0	7　　　0	7　　　0
相対値	3	4	～	11	12
TAB位置	0　　　7	8　　　15	16　～　63	64　　　71	72　　　79

復帰情報　このコマンドはエラーを生じません．したがって復帰情報はありません．

サンプル　入力された位置にＴＡＢ位置を設定します．８０以上を入力すると終了します．

```
PAGE   001 (          ,          )

01000                              *
01010                              * SUBSYSTEM TABSET
01020                              *
01030                                     OPT    M,NOS,NOG,NOO,PAGE=255
01040                    9BDB      MESSAG EQU    $9BDB    MESSAGE OUT FROM X+1
01050                    D08E      OUT    EQU    $D08E    ONE CHR OUT
01060                    FBFA      BIOS   EQU    $FBFA    BIOS (EXTEND-INDIRECT)
01070                    DB54      KEYIN  EQU    $DB54    KEYINPUT WITH WAIT
01080                              *
01090  5000                               ORG    $5000
01100                    5000      START  EQU    *
01110                              *
01120                              * tabulation point set
01130                              *
01140  5000 31          8D 00D2   TABSET LEAY   TABPOS,PCR tabset bitmap addr.
01150  5004 5F                            CLRB            clear bitmap
01160  5005 6F          A5        T01    CLR    B,Y
01170  5007 5C                            INCB
01180  5008 C1          0A                CMPB   #10
```

```
01190  500A 26    F9   5005          BNE    T01
01200  500C 30    8D 0084    T02     LEAX   MES1-1,PCR
01210  5010 BD    9BDB               JSR    MESSAG
01220  5013 8D    60   5075          BSR    GETNUM   get tabposition
01230  5015 31    8D 00BD            LEAY   TABPOS,PCR
01240  5019 C1    50                 CMPB   #80      Pos>80 then end
01250  501B 24    17   5034          BHS    T05
01260  501D 1F    98                 TFR    B,A      A=B
01270  501F 44                       LSRA            A=A/8
01280  5020 44                       LSRA
01290  5021 44                       LSRA
01300  5022 31    A6                 LEAY   A,Y      get pos addr.
01310  5024 86    80                 LDA    #$80
01320  5026 C4    07                 ANDB   #$07     B=B MOD 8
01330  5028 27    04   502E          BEQ    T04
01340  502A 44               T03     LSRA            get on-bit
01350  502B 5A                       DECB
01360  502C 26    FC   502A          BNE    T03
01370  502E AA    A4         T04     ORA    ,Y       set tabposition
01380  5030 A7    A4                 STA    ,Y
01390  5032 20    D8   500C          BRA    T02
01400                        *
01410                        * display tabulation point map
01420                        *
01430  5034 30    8D 007D    T05     LEAX   MES2-1,PCR end of tabset
01440  5038 BD    9BDB               JSR    MESSAG   bit_map display
01450  503B 5F                       CLRB
01460  503C A6    A5         T06     LDA    B,Y      get bit_map
01470  503E 34    04                 PSHS   B        output 8 bit
01480  5040 C6    08                 LDB    #8
01490  5042 49               T07     ROLA
01500  5043 34    02                 PSHS   A
01510  5045 86    2E                 LDA    #'.      if not tabposition
01520  5047 24    02   504B          BCC    T08
01530  5049 86    2A                 LDA    #'*      if tabposition
01540  504B BD    D08E       T08     JSR    OUT
01550  504E 35    02                 PULS   A
01560  5050 5A                       DECB
01570  5051 26    EF   5042          BNE    T07
01580  5053 35    04                 PULS   B
01590  5055 5C                       INCB
01600  5056 C1    0A                 CMPB   #10      10 bytes end ?
01610  5058 26    E2   503C          BNE    T06
01620                        *
01630                        * execute subsys-command TABSET
01640                        *
01650  505A 31    3D                 LEAY   -3,Y     get subsys-command addr.
01660  505C 86    0B                 LDA    #$0B     subsys-command TABSET
01670  505E A7    22                 STA    2,Y      set subsys-command
01680  5060 30    8D 007C            LEAX   RCB,PCR get rcb addr.
01690  5064 86    10                 LDA    #$10     bios request
01700  5066 A7    84                 STA    ,X
01710  5068 10AF  02                 STY    2,X      set subsys-command addr
01720  506B CC    000D               LDD    #13      transfer 13 bytes (=3+10)
01730  506E ED    04                 STD    4,X
01740  5070 AD    9F FBFA            JSR    [BIOS]   execute command
01750  5074 39                       RTS
01760                        *
01770                        * subroutine:get number 0..255 in Breg
01780                        *
01790  5075 5F               GETNUM  CLRB            get number(0-255) in Breg
01800  5076 4F                       CLRA
01810  5077 34    02         GN01    PSHS   A
01820  5079 86    0A                 LDA    #10      Breg=Breg*10
01830  507B 3D                       MUL
01840  507C EB    E0                 ADDB   ,S+      Breg=Breg+keyinput
01850  507E 34    04                 PSHS   B
01860  5080 BD    DB54               JSR    KEYIN
01870  5083 35    04                 PULS   B
01880  5085 81    30                 CMPA   #'0      TEST '0'..'9'
01890  5087 25    0B   5094          BCS    GN02
01900  5089 81    39                 CMPA   #'9
01910  508B 22    07   5094          BHI    GN02
01920  508D BD    D08E               JSR    OUT
```

```
01930  5090 80   30                 SUBA   #'0
01940  5092 20   E3   5077          BRA    GN01
01950  5094 39              GN02    RTS            num.0-255 in Breg
01960                       *
01970                       * messages
01980                       *
01990  5095      0D0A       MES1    FDB    $0D0A
02000  5097      54                 FCC    'TAB position(0-79),(end=80) ?
02010  50B5      00                 FCB    0
02020  50B6      0D0A       MES2    FDB    $0D0A
02030  50B8      3C                 FCC    '<<<  TABSET bit_map  >>>'
02040  50D0      0D0A               FDB    $0D0A
02050  50D2      00                 FCB    0
02060                       *
02070                       * working area
02080                       *
02090  50D3      0003       SUBCOM RMB     3       for subsys-command
02100  50D6      000A       TABPOS RMB     10      tabset bit map
02110  50E0      0006       RCB     RMB    6       for bios
02120                       *
02130            5000               END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50E5
PROGRAM ENTRY ADDR=5000


10 ' TABSET sample program
20 FOR I=1 TO 20
30 PRINT CHR$(9); :' tab
40 PRINT "!";     :' here is tabposition
50 NEXT I
60 END



exec &H5000

TAB position(0-79),(end=80) ? 0
TAB position(0-79),(end=80) ? 20
TAB position(0-79),(end=80) ? 30
TAB position(0-79),(end=80) ? 50
TAB position(0-79),(end=80) ? 70
TAB position(0-79),(end=80) ? 75
TAB position(0-79),(end=80) ? 80
<<<  TABSET bit_map  >>>
*...................*.........*...................*...................*....*....

Ready
run
                    !         !                   !                   !    !
!                   !         !                   !                   !    !
!                   !         !                   !                   !    !
!                   !         !
Ready
```

SUB：CNSCTL　―コンソール機能設定―

機　能　サブシステムのコンソール機能の設定を行います．

パラメータ

0	＄00	
1	＄00	
2	＄0C	コマンドコード
3	CF	機能設定フラグ　　1＝選択　0：非選択

bit 5　：CR（＄0D）が出力された時に同時にLF（＄0A）動作を行う．

bit 4　：文字列出力時にESCキーによる表示の停止を有効にする．

bit 3　：ページモード画面において，文字がいっぱいになったときに，ベルをならして表示動作を停止しキー入力を待つ．(ページリミット動作)

bit 2　：TAB動作を有効にする．

bit 1　：オーダ動作を有効にする．

bit 0　：カーソルを表示する．

復帰情報　なし．(エラーは生じません．)

解　説　リセット時には，機能設定フラグは＄23（100011_2）に設定されます．

オーダ動作を有効にしなかった場合(bit 1＝0)，コードが＄00～＄1F及び＄7Fのキャラクタは文字として表示されます．

サンプル　入力にしたがって機能を設定します．オーダ動作を有効としなかった場合には，BASICのコマンドは入力できなくなるので注意して下さい．(Abortすれば回復します)．

またBASICでは改行を行う時に＄0D，＄0Aを両方出力しているので，bit 5をONにした場合には，みかけ上2行改行します．

```
01000                          *
01010                          * SUBSYSTEM CNSCTL
01020                          *
01030                                 OPT    M,NOS,NOG,NOO,PAGE=255
01040             9BDB         MESSAG EQU    $9BDB    message out from X+1
01050             D0BE         OUT    EQU    $D0BE    one chr out
01060             FBFA         BIOS   EQU    $FBFA    bios (extend-indirect)
01070             DB54         KEYIN  EQU    $DB54    keyinput with wait
01080                          *
01090  5000                           ORG    $5000
01100             5000         START  EQU    *
01110                          *
01111                          * set parameters
01112                          *
01120  5000 31    8D 0116      CNSCTL LEAY   SUBCOM,PCR
01130  5004 6F    23                  CLR    3,Y      clear flags
01140  5006 33    8D 003E             LEAU   MESPTR,PCR for message output
01150  500A C6    20                  LDB    #%00100000 counter
01160  500C 34    04           C01    PSHS   B
01170  500E AE    C1                  LDX    ,U++     load message address
01180  5010 1F    50                  TFR    PC,D
01190             5013         OFFSET EQU    *+1
01200  5012 30    8B                  LEAX   D,X      add offset
01210  5014 34    40                  PSHS   U
01220  5016 BD    9BDB                JSR    MESSAG   message output
01230  5019 BD    DB54                JSR    KEYIN    keyin
01240  501C BD    D0BE                JSR    OUT      echo back
01250  501F 35    40                  PULS   U
01260  5021 E6    E4                  LDB    ,S       load counter
01270  5023 84    01                  ANDA   #$01     keyinput='1' ?
01280  5025 27    04   502B           BEQ    C02
01290  5027 EA    23                  ORB    3,Y      flag on !
01300  5029 E7    23                  STB    3,Y
01310  502B 35    04           C02    PULS   B        counter come back
01320  502D 54                        LSRB            count up
01330  502E 26    DC   500C           BNE    C01
01331                          *
01332                          * subsys-command CNSCTL execution
01333                          *
01340  5030 86    0C                  LDA    #$0C     subsys-command CNSCTL
01350  5032 A7    22                  STA    2,Y      set command
01360  5034 30    8D 000A             LEAX   RCB,PCR  get rcb address
01370  5038 AD    9F FBFA             JSR    [BIOS]   command execution
01380  503C A6    23                  LDA    3,Y      load flags
01390  503E B7    05A8                STA    $5A8     initialize for BASIC flags
01400  5041 39                        RTS
01410                          *
01411                          * constant
01412                          *
01420  5042       1000         RCB    FDB    $1000,SUBCOM,4 for bios
01430                          *
01440  5048       0041         MESPTR FDB    MES1-OFFSET for message output
01450  504A       0062                FDB    MES2-OFFSET
01460  504C       0083                FDB    MES3-OFFSET
01470  504E       00A4                FDB    MES4-OFFSET
01480  5050       00C5                FDB    MES5-OFFSET
01490  5052       00E6                FDB    MES6-OFFSET
01500                          *
01501                          * messages
01502                          *
01510             0D0A         CRLF   EQU    $0D0A    output messages
01520  5054       0D0A         MES1   FDB    CRLF
01530  5056       61                  FCC    'auto LF ?         (ON:1,OFF:0)  '
01540  5074       00                  FCB    0
01550  5075       0D0A         MES2   FDB    CRLF
01560  5077       70                  FCC    'put wait ?        (ON:1,OFF:0)  '
01570  5095       00                  FCB    0
01580  5096       0D0A         MES3   FDB    CRLF
01590  5098       70                  FCC    'page wait ?       (ON:1,OFF:0)  '
01600  50B6       00                  FCB    0
01610  50B7       0D0A         MES4   FDB    CRLF
01620  50B9       54                  FCC    'TAB action ?      (ON:1,OFF:0)  '
```

```
01630  50D7     00               FCB    0
01640  50D8     0D0A     MES5    FDB    CRLF
01650  50DA     6F               FCC    'order action ?   (ON:1,OFF:0)
01660  50F8     00               FCB    0
01670  50F9     0D0A     MES6    FDB    CRLF
01680  50FB     63               FCC    'cursor display ? (ON:1,OFF:0)
01690  5119     00               FCB    0
01700                    *
01701                    * working area
01702                    *
01710  511A     0004     SUBCOM RMB     4        for subsys-command
01715                    *
01720           5000             END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=511D
PROGRAM ENTRY ADDR=5000
```

```
exec &H5000

auto LF ?          (ON:1,OFF:0) 0
put wait ?         (ON:1,OFF:0) 1
page wait ?        (ON:1,OFF:0) 1
TAB action ?       (ON:1,OFF:0) 1
order action ?     (ON:1,OFF:0) 1
cursor display ?   (ON:1,OFF:0) 1
Ready
```

ＳＵＢ：ＬＩＮＥ　―直線又は四角の表示―

機　能　ＢＡＳＩＣのＬＩＮＥ文に相当し，直線又は四角を表示します．

パラメータ

0	$00	
1	$00	
2	$15	コマンドコード
3	CL	カラーコード（０～７）
4	F	ファンクションコード（ＮＯＴを除く，０～４）
5 6	X1	始点のＸ座標（０～６３９）
7 8	Y1	始点のＹ座標（０～１９９）
9 10	X2	終点のＸ座標（０～６３９）
11 12	Y2	終点のＹ座標（０～１９９）
13	B	ボックスフラグ　０：直線表示 １：四角形の枠表示 ２：四角形ぬりつぶし表示

復帰情報

0	E	エラーコード

解　説　四角形表示のときには（Ｘ１，Ｙ１）－（Ｘ２，Ｙ２）を対角線とする四角形を表示します．

注）グラフィック関係のサンプルはまとめて示します．

SUB：CHAIN　—連続直線表示—

機　能　指定された座標を順々に直線で結びます．BASICのCONNECT文に相当します．

パラメータ

0	$00	
1	$00	
2	$16	コマンドコード
3	CL	カラーコード（0～7）
4	F	ファンクションコード（0～4，NOTを除く）
5	N	座標数（2～30）
6、7	X1	X座標（0～639）
8、9	Y1	Y座標（0～199）
10、11	X2	
12、13	Y2	
∫	≈	
4・N+2、3	Xn	
4・N+4、5	Yn	

復帰情報

0	E	エラーコード

ＳＵＢ：ＰＯＩＮＴ　－点表示－

機　能　指定された座標に点を表示します．

パラメータ

オフセット	内容	説明
0	$00	
1	$00	
2	$17	コマンドコード
3	N	表示点数（1～20）
4、5	X1	X座標（0～639）
6、7	Y1	Y座標（0～199）
8	CL1	カラーコード（0～7）
9	F1	ファンクションコード（0～4，NOTを除く）
∫	≈	
6N－2,1	Xn	
6N＋0,1	Yn	
6N＋2	CLn	
6N＋3	Fn	

復帰情報

オフセット	内容	説明
0	E	エラーコード

SUB：PAINT —ペイント—

機　能 指定された座標を含み，境界色によって囲まれた部分を，指定色で塗りつぶします．

パラメータ

0	$ 00	
1	$ 00	
2	$ 18	コマンドコード
3–4	X	X座標（0～639）
5–6	Y	Y座標（0～199）
7	PC	ペイントカラーコード（0～7）
8	NC	境界色の数（1～8）
9	BC1	境界色（0～7）
～	～	
NC～8	BCn	

復帰情報

0	E	エラーコード

解　説 ペイントカラー及び画面の端も，境界色とみなされます．

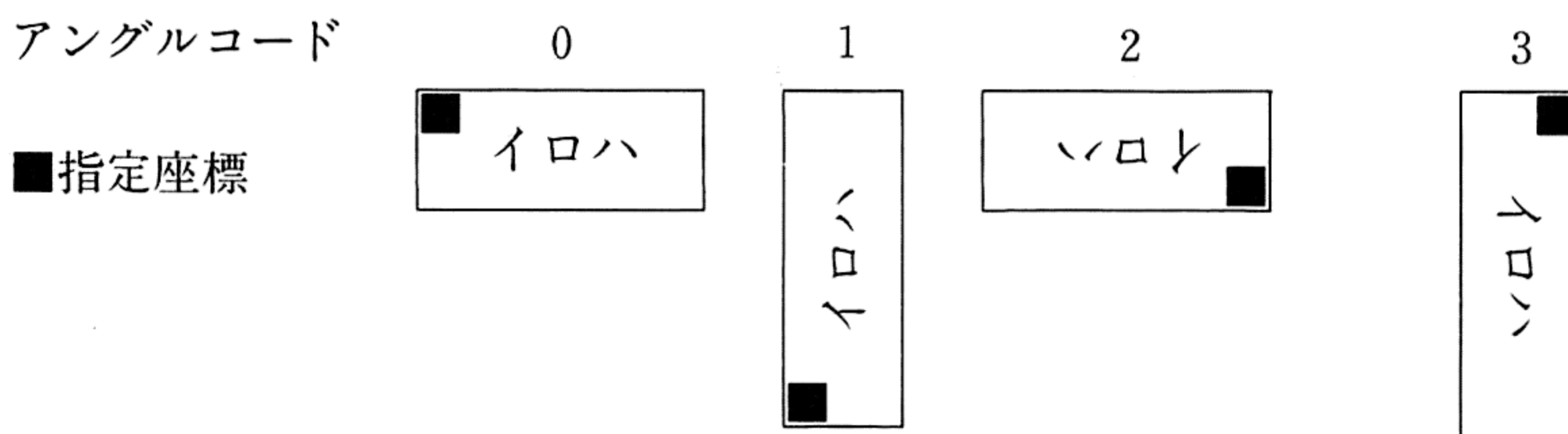
SUB：SYMBOL —文字列表示—

機　能　文字列を，指定した角度，大きさで表示します．

パラメータ

0	$00	
1	$00	
2	$19	コマンドコード
3	CL	カラーコード（0～7）
4	F	ファンクションコード（0～5）
5	A	アングルコード（下図参照，0～3）
6	W	文字横幅倍率（0～255）
7	H	文字縦幅倍率（0～255）
8 9	X	X座標（0～639）
10 11	Y	Y座標（0～199）
12	N	文字数（1～80）
13 ～ N＋12	String	文字列

アングルコード　0　1　2　3

■指定座標

復帰情報

0	E	エラーコード

解　説　倍率は，1，1で桁数80字のときのキャラクタと同じ大きさになります．

SUB：CCOLOR　—枠内色変換—

機　能　枠内の指定色を他の色へ変更します.

パラメータ

0	$00	
1	$00	
2	$1A	コマンドコード
3，4	X1	開始位置　X座標（0～639）
5，6	Y1	Y座標（0～199）
7，8	X2	終了位置　X座標（0～639）
9，，0	Y2	Y座標（0～199）
11	N	変更するカラーの数（1～8）
12	CLA1	旧カラーコード（0～7）
13	CLB1	新カラーコード（0～7）
≈		
2N+10	CLAn	
2N+11	CLBn	

復帰情報

0	E	エラーコード

解　説　このコマンドは，枠内の旧カラーコードのドットをすべて新カラーコードのドットにおきかえます.

このコマンドは，F－BASICでは使用されておらずパレットによる色のおきかえとは異なります.

ＳＵＢ：ＧＧＢＬＫ１　―枠内ドット読み取り―

機　能　枠内のドットをビットパターンとして読み取ります．

パラメータ

0	$00	
1	$00	
2	$1B	コマンドコード
3、4	X1	開始位置　X座標（0～639）
5、6	Y1	Y座標（0～199）
7、8	X2	終了位置　X座標（0～639）
9、10	Y2	Y座標（0～199）
11	N	読み取る色の数（1～8）
12	CL1	カラーコード（0～7）
～		
N＋11	CLn	

復帰情報

0	E	エラーコード
1	S'	継続フラグ（bit7）
2		
3	B	転送バイト数
4	Pattern	画面データ
～		
B＋3		

継続フラグがONのときは，ＣＯＮＴコマンドで残りのデータを要求する必要があります．指定色のドットのみを転送します．（指定色は８色まで）

ドットパターンと画面データとの対応を下図に示します．

ドットパターン

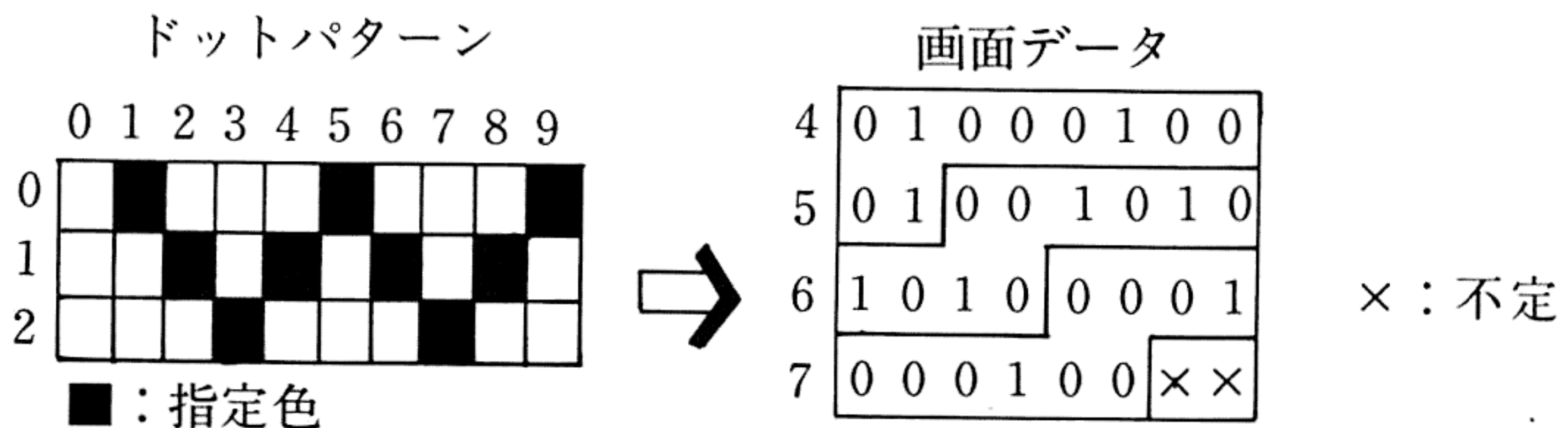
	0	1	2	3	4	5	6	7	8	9
0	□	■	□	□	□	■	□	□	□	■
1	□	□	■	□	■	□	■	□	■	□
2	□	□	□	■	□	□	□	■	□	□

■：指定色

⇨

画面データ

4	0 1 0 0 0 1 0 0
5	0 1 0 0 1 0 1 0
6	1 0 1 0 0 0 0 1
7	0 0 0 1 0 0 × ×

×：不定

ＳＵＢ：ＰＧＢＬＫ１　―枠内ドット表示―

機　能　ＧＧＢＬＫ１と逆に枠内にドットパターンを指定色で表示します．

パラメータ

0	$00	
1	S'	継続フラグ（bit 7）
2	$1C	コマンドコード
3、4、	X1	開始位置　X座標（0～639）
5、6、	Y1	Y座標（0～199）
7、8	X2	終了位置　X座標（0～639）
9、10	Y2	Y座標（0～199）
11	CL	カラーコード（0～7）
12	F	ファンクションコード（0～5）
13	B	転送バイト数（1～114）
14	Pattern	画面データ
∫		
B＋13		

分割して転送する場合は継続フラグを１にしてコマンドを送り，残りのテーダをＣＯＮＴコマンドで転送する必要があります．

復帰情報

0	E	エラーコード

解　説　ドットパターンと画面データとの対応はＧＧＢＬＫ１と同様です．

SUB：GGBLK２　―枠内ドット読み取り(色付き)―

機　能　枠内のドットをビットパターンとして読み取ります．この際，ＲＧＢのすべての画面を読み取ります．

パラメータ

0	$00		
1	$00		
2	$1D	コマンドコード	
3、4	X1	開始位置	X座標（0～639）
3、6	Y1		Y座標（0～199）
7、8	X2	終了位置	X座標（0～639）
9、10	Y2		Y座標（0～199）

復帰情報

0	E	エラーコード
1	S'	継続フラグ（bit 7）
2		
3	B	転送バイト数
4	Pattern	画面データ
∫		
B＋3		

継続フラグがＯＮのときは，ＣＯＮＴコマンドで残りのデータを要求する必要があります．

画面のドットパターンは，各原色ごとに，青・赤・緑の順で転送されます．それぞれの原色でのドットパターンと画面データとの対応はＧＧＢＬＫ１と同様です．

ＳＵＢ：ＰＧＢＬＫ２　―枠内ドット表示(色付き)―

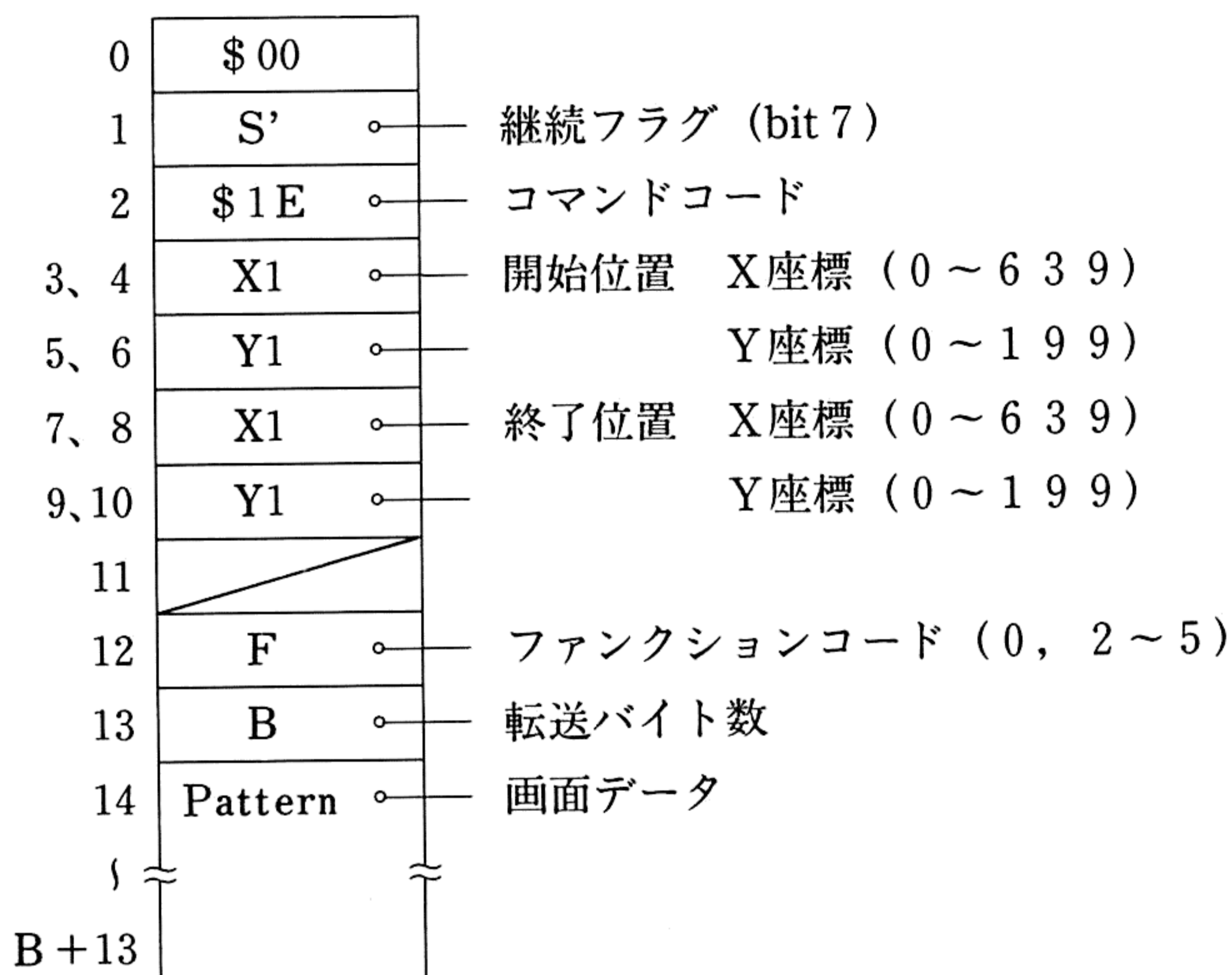
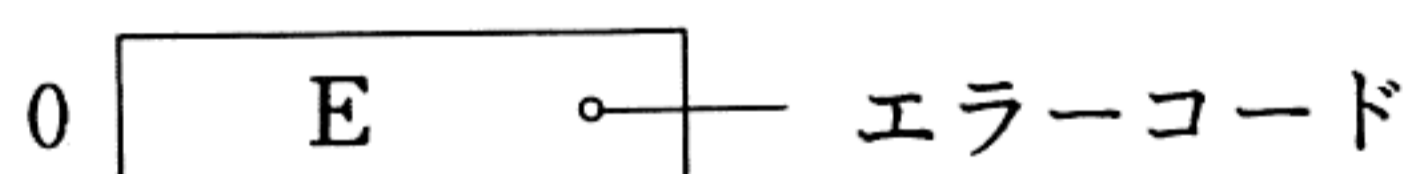
機　能　ＧＧＢＬＫ２と逆に枠内にドットパターンを表示します．この場合ドット毎の色はＧＧＢＬＫ２で読み込んだ時と同様になります．

パラメータ

0	$00	
1	S'	継続フラグ（bit 7）
2	$1E	コマンドコード
3、4	X1	開始位置　X座標（０～６３９）
5、6	Y1	Y座標（０～１９９）
7、8	X1	終了位置　X座標（０～６３９）
9、10	Y1	Y座標（０～１９９）
11		
12	F	ファンクションコード（０，２～５）
13	B	転送バイト数
14	Pattern	画面データ
～		
B＋13		

分割して転送する場合は継続フラグを１にしてコマンドを送り，残りのデータをＣＯＮＴコマンドで転送します．

復帰情報

0	E	エラーコード

解　説　ファンクションコードにＰＲＥＳＥＴ（1）は設定できません．各原色ごとのドットパターンと画面データとの対応はＧＧＢＬＫ１と同様です．転送の順序は青・赤・緑の順です．

ＳＵＢ：ＧＣＵＲＳ　―座標読み取り―

機　能　グラフィックカーソルを使用してオペレータが示した座標を，読み取ります．座標数は１０個まで可能です．

パラメータ

0	$ 00	
1	$ 00	
2	$1F	コマンドコード
3	CL	グラフィックカーソルの色（０～７）
4	N	読み取る座標の数（１～１０）
5、6	X	初期座標　X座標（０～６３９）　最初にグラフィックカーソルを表示する座標
7、8	Y	Y座標（０～１９９）

復帰情報

0	E	エラーコード
1		
2		
3、4	X1	読み取った座標　X座標
5、6	Y1	Y座標
～	～	
4N－1.0	Xn	
4N＋1.2	Yn	

解　説　グラフィックカーソルの移動の方法はＦ－ＢＡＳＩＣと同様です．

このコマンドは，読み取る座標の数だけ座標を得る（Return キーが押される）まで終了しません．

ＳＵＢ：ＣＬＩＮＥ　―文字による直線又は四角の表示―

機　能　文字によって直線や四角を表示します．

パラメータ

0	$00	
1	$00	
2	$20	コマンドコード
3	CL	カラーコード（０～７）
4	CHR	文字コード（＄２０～＄ＦＥ）
5		
6	X1	始点のX座標（０～７９）
7		
8	Y1	始点のY座標（０～２４）
9		
10	X2	終点のX座標（０～７９）
11		
12	Y2	終点のY座標（０～２４）
13	B	ボックスフラグ　０：直線表示 １：四角形の枠を表示 ２：四角形ぬりつぶし

復帰情報

0	E	エラーコード

解　説　座標の指定はキャラクタ座標なので注意して下さい．指定した文字（ＣＨＲ）で直線ないし四角を表示します．

〈グラフィック関係サブシステムコマンド・サンプルプログラム〉

グラフィック関係のサブシステムコマンドを用いたサンプルを示します．ここで使用したコマンドは，

ＥＲＡＳＥ，ＬＩＮＥ，ＣＨＡＩＮ，ＰＡＩＮＴ，ＳＹＭＢＯＬ
ＣＣＯＬＯＲ

の６つです．

サンプル

```
PAGE  001 (        ,000001)

01000                                *
01010                                * subsystem GRAPHIC command sample
01020                                *
01030                                *  !!! not position independent !!!
01040                                *
01050                                       OPT     M,NOS,NOG,PAGE=255
01060                FBFA      BIOS     EQU     $FBFA    BIOS (extend-indirect)
01070                                *
01080   5000                               ORG     $5000
01090                5000      START    EQU     *
01100                                *
01110                                * MAIN PROGRAM
01120                                *
01130   5000 8E      500F               LDX     #RCBPTR load rcb address
01140   5003 AD      9F FBFA   M01      JSR     [BIOS]  call bios
01150   5007 30      06                 LEAX    6,X     load next rcb address
01160   5009 8C      5051               CMPX    #RCBEND end ?
01170   500C 26      F5   5003          BNE     M01
01180   500E 39                         RTS
01190                                *
01200                                * RCB for subsystem call
01210                                *
01220                500F      RCBPTR EQU      *
01230   500F         1000               FDB     $1000,SUBC0,SUBC1-SUBC0
01240   5015         1000               FDB     $1000,SUBC1,SUBC2-SUBC1
01250   501B         1000               FDB     $1000,SUBC2,SUBC3-SUBC2
01260   5021         1000               FDB     $1000,SUBC3,SUBC4-SUBC3
01270   5027         1000               FDB     $1000,SUBC4,SUBC5-SUBC4
01280   502D         1000               FDB     $1000,SUBC5,SUBC6-SUBC5
01290   5033         1000               FDB     $1000,SUBC6,SUBC7-SUBC6
01300   5039         1000               FDB     $1000,SUBC7,SUBC8-SUBC7
01310   503F         1000               FDB     $1000,SUBC8,SUBC9-SUBC8
01320   5045         1000               FDB     $1000,SUBC9,SUBC10-SUBC9
01330   504B         1000               FDB     $1000,SUBC10,SUBC11-SUBC10
01340                5051      RCBEND EQU      *
01350                                *
01360                                * subsystem commands
01370                                *
01380                                *      ERASE
01390   5051         00        SUBC0    FCB     0,0,$02,0,0
01400                                *      SYMBOL
01410   5056         0000      SUBC1    FDB     0,$1905,0,$0303,20,0
01420   5062         19                 FCB     SUBC2-MES
01430   5063         46        MES      FCC     'Fujitsu Personal Computer'
01440                                *      LINE (B)
01450   507C         0000      SUBC2    FDB     0,$1507
01460   5080         00                 FCB     0
01470   5081         0000               FDB     0,0,639,199,$0100
01480                                *      CHAIN for 'F'
01490   508B         0000      SUBC3    FDB     0,$1606,11
01500   5091         0014               FDB     20,30,20,170,50,170,50,110
01510   50A1         0096               FDB     150,110,150,90,50,90,50,50
01520   50B1         00BE               FDB     190,50,190,30,20,30
01530                                *      CHAIN for 'M'
01540   50BD         0000      SUBC4    FDB     0,$1606,13
01550   50C3         00DC               FDB     220,30,220,170,250,170,250,80
01560   50D3         0140               FDB     320,150,390,80,390,170,420,170
01570   50E3         01A4               FDB     420,30,390,30,320,100,250,30
01580   50F3         00DC               FDB     220,30
01590                                *      CHAIN for '7'
01600   50F7         0000      SUBC5    FDB     0,$1606,10
01610   50FD         01C2               FDB     450,30,450,70,480,70,480,50
01620   510D         024E               FDB     590,50,480,170,510,170,620,50
01630   511D         026C               FDB     620,30,450,30
01640                                *      PAINT for 'F'
01650   5125         00        SUBC6    FCB     0,0,$18
01660   5128         0016               FDB     22,32
01670   512C         02                 FCB     2,1
01680   512E         06                 FCB     6
01690                                *      PAINT for 'M'
```

```
01700  512F      00        SUBC7  FCB    0,0,$18
01710  5132      0140             FDB    320,148
01720  5136      02               FCB    2,1
01730  5138      06               FCB    6
01740                      *         PAINT for '7'
01750  5139      00        SUBC8  FCB    0,0,$18
01760  513C      01C4             FDB    452,32
01770  5140      02               FCB    2,1
01780  5142      06               FCB    6
01790                      *         CCOLOR
01800  5143      00        SUBC9  FCB    0,0,$1A
01810  5146      0000             FDB    0,0,639,199
01820  514E      05               FCB    5
01830  514F      00               FCB    0,1,2,4,5,2,6,7,7,5
01840                      *         CCOLOR
01850  5159      00        SUBC10 FCB    0,0,$1A
01860  515C      0000             FDB    0,0,639,199
01870  5164      05               FCB    5
01880  5165      01               FCB    1,0,4,1,2,3,7,5,5,7
01890            516F      SUBC11 EQU    *
01900                      *
01910            5000             END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=516E
PROGRAM ENTRY ADDR=5000
```

Fujitsu Personal Computer

FM7

ＳＵＢ：ＩＮＫＥＹ　ーキー入力ー

機　　能　キーボートから１文字取り込みます．

パラメータ

$ 00
$ 00
$ 29 ○── コマンドコード
CF ○── 制御フラグ

bit 0：ウェイトフラグ
bit 1：Resetフラグ

ウエイトフラグを１にすると，キー入力があるまで待ちます．また，ＲＥＳＥＴフラグを１にすると，キー入力に先だってキー入力バッファ（ＳＵＢＣＰＵメモリ内にある.)をクリアします．

復帰情報

0	$ 00
1	
2	
3	K ○── 入力されたキーのアスキーコード(0～255)
4	EN ○── 入力の有無 0：なし 1：あり

解　　説　ＥＮが０のときのＫの値は意味を持ちません．

オーダによってキーの先行入力を可能にしている場合には，先行入力した文字が順々にセットされます．よってＦＭ７では，ＰＣやＭＺなどのようにキーがその時点で押されていないという状態を得る（本当の意味でのリアルタイムキー入力）ことはできません．

ＦＭ７のこのコマンドでは，ＦＭ８のこのコマンドにあった制御フラグのうちbit 2のＲＥＳＥＴ２のフラグがなくなっています．これはサブＣＰＵのメモリ上にキー入力バッファ（32文字分）が作られたために削除されたものです．

サンプル　ＧＢＡＤＲの項を参照（ＧＢＡＤＲのサンプルで使用）

SUB：DEFPF　—PFキー定義—

機　能　プログラマブル・ファンクションキーに文字列を定義します．

パラメータ

0	$00	
1	$00	
2	$2A	コマンドコード
3	NO	PFキー番号（1～10）
4	N	文字数（0～15）
5	String	文字列
⁝	⁝	
4+N		

復帰情報

0	E	エラーコード

解　説　ファンクションキーが割り込み用に定義されている時には，このコマンドにより文字列は定義されますが，PFキーを押しても期待する文字列は得られません．しかし，割り込み用の定義を解除すれば，最後に設定された文字列が得られるようになります．

SUB：GETPF　－PFキー文字列読み取り－

機　　能　プログラマブル・ファンクションキーに定義されている文字列を読み取ります.

パラメータ

0	$00	
1	$00	
2	$2B	コマンドコード
3	NO	PFキー番号（1～10）

復帰情報

0	$00	
1		
2		
3	NO	PFキー番号(1～10)
4	N	文字数(0～15)
5	String	文字例
～		
4＋N		

解　　説　このコマンドでは，ＰＦキーが割り込み用に設定されているか否かにかかわらず，ＰＦキーに設定されている文字列を返します.

割り込み用として設定されている場合には，ＰＦキーを押しても，このコマンドで返される文字列は得られません．割り込み設定が解除されれば，文字列が得られるのは，ＤＥＦＰＦの項で述べた通りです.

サンプル

```
PAGE  001 (          ,          )

01000                          *
01010                          * SUBSYSTEM DEFPF & GETPF
01020                          *
01030                                  OPT     M,NOO,NOS,NOG,P=255
01040            FBFA          BIOS    EQU     $FBFA    bios (extend-indirect)
01050            9BDB          MESSAG  EQU     $9BDB    output from (X+1) till 0
01060            DB54          KEYIN   EQU     $DB54    keyinput with wait
01070            D08E          OUT     EQU     $D08E    one character output
01080            D807          LINEIN  EQU     $D807    line input
01090                          *
```

```
01100  5000                        ORG     $5000
01110             5000      START  EQU     *
01120                       *
01130                       * set PFkey number
01140                       *
01150  5000 30    8D 0082   PFNUM  LEAX    MES1-1,PCR output message
01160  5004 BD    9BDB             JSR     MESSAG
01170  5007 BD    DB54             JSR     KEYIN      keyinput
01180  500A 81    30               CMPA    #'0        test '0'..'9'
01190  500C 25    78   5086        BCS     END
01200  500E 81    39               CMPA    #'9
01210  5010 22    74   5086        BHI     END
01220  5012 BD    D08E             JSR     OUT        echoback
01230  5015 80    30               SUBA    #'0
01240  5017 26    02   501B        BNE     GETPF
01250  5019 86    0A               LDA     #10        '0'-->'10'
01260                       *
01270                       * get PFkey string
01280                       *
01290  501B 31    8D 00AB   GETPF  LEAY    SUBCOM,PCR
01300  501F A7    23               STA     3,Y        set PFkey no.
01310  5021 86    2B               LDA     #$2B       subsys-command GETPF
01320  5023 A7    22               STA     2,Y        set command
01330  5025 30    8D 0099          LEAX    RCB,PCR
01340  5029 86    11               LDA     #$11       bios SUBIN
01350  502B A7    84               STA     ,X
01360  502D 10AF  02               STY     2,X        set command addr.
01370  5030 CC    0014             LDD     #20        set command length
01380  5033 ED    04               STD     4,X
01390  5035 ED    06               STD     6,X
01400  5037 AD    9F FBFA          JSR     [BIOS]     command executoin
01410  503B 30    8D 0060          LEAX    MES2-1,PCR output message
01420  503F BD    9BDB             JSR     MESSAG
01430  5042 E6    24               LDB     4,Y        output PFkey string
01440  5044 CB    05               ADDB    #5
01450  5046 6F    A5               CLR     B,Y
01460  5048 30    24               LEAX    4,Y
01470  504A BD    9BDB             JSR     MESSAG
01480                       *
01490                       * def PFkey string
01500                       *
01510  504D 30    8D 005F          LEAX    MES3-1,PCR output message
01520  5051 BD    9BDB             JSR     MESSAG
01530  5054 BD    D807             JSR     LINEIN     line input
01540  5057 30    01               LEAX    1,X        X=X+1:X=string top addr.
01550  5059 33    25               LEAU    5,Y        PFkey string area top
01560  505B 5F                     CLRB               counter clear
01570  505C A6    80        DO1    LDA     ,X+        set PFkey string
01580  505E 27    09   5069        BEQ     DO2        end of string ?
01590  5060 A7    C0               STA     ,U+
01600  5062 C1    0F               CMPB    #15        over 15 chr ?
01610  5064 27    03   5069        BEQ     DO2
01620  5066 5C                     INCB
01630  5067 20    F3   505C        BRA     DO1
01640  5069 E7    24        DO2    STB     4,Y        set PFkey string length
01650  506B 86    2A               LDA     #$2A       subsys-command DEFPF
01660  506D A7    22               STA     2,Y        set command
01670  506F 30    8D 004F          LEAX    RCB,PCR
01680  5073 86    10               LDA     #$10       bios SUBOUT
01690  5075 A7    84               STA     ,X
01700  5077 10AF  02               STY     2,X        set command addr.
01710  507A CC    0014             LDD     #20        set command length
01720  507D ED    04               STD     4,X
01730  507F AD    9F FBFA          JSR     [BIOS]     command execution
01740  5083 16    FF7A 5000        LBRA    PFNUM
01750                       *
01760                       * end of execution
01770                       *
01780  5086 39              END    RTS
01790                       *
01800                       * messages
01810                       *
01820  5087       0D0A      MES1   FDB     $0D0A
01830  5089       50               FCC     'PFkey number (1..9,0)='
```

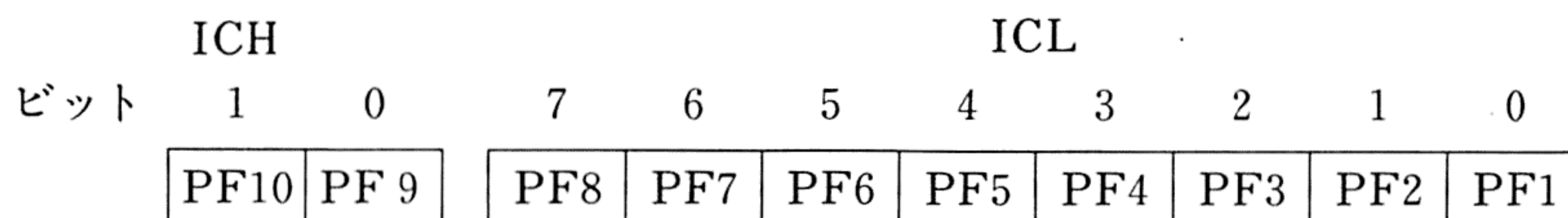
```
01840  509F      00                FCB    0
01850  50A0      0D0A      MES2    FDB    $0D0A
01860  50A2      20                FCC    '      String ='
01870  50AE      11                FCB    $11,$07,0 set field top
01880  50B1      0D0A      MES3    FDB    $0D0A
01890  50B3      4E                FCC    'New String ='
01900  50BF      11                FCB    $11,$07,0 set field top
01910                      *
01920                      * working area
01930                      *
01940  50C2      0008      RCB     RMB    8
01950  50CA      0014      SUBCOM  RMB    20
01960                      *
01970            5000              END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50DD
PROGRAM ENTRY ADDR=5000


 Ready
 exec &H5000

 PFkey number (1..9,0)=1
      String =AUTO
 New String =abcdef12345abcdef

 PFkey number (1..9,0)=
 Ready
 keylist

 PF1      abcdef12345abcd
 PF2      LIST
 PF3      RUN
 PF4      CONT
 PF5      LLIST
 PF6      LOAD
 PF7      SAVE"
 PF8      ?DATE$,TIME$
 PF9      SCREEN7,7
 PF10     HARDC

Ready
HARDC
```

ＳＵＢ：ＩＮＴＣＴＬ　―ＰＦキー割り込み定義―

機　能　**プログラマブル・ファンクションキーを割り込み用に定義します．**

パラメータ

0	＄00
1	＄00
2	＄2C　―コマンドコード
3	ICH　―割り込み制御フラグ
4	ICL

	ICH		ICL							
ビット	1	0	7	6	5	4	3	2	1	0
	PF10	PF 9	PF8	PF7	PF6	PF5	PF4	PF3	PF2	PF1

’1’になっているビットのＰＦキーが割り込み用に定義されます．割り込み用に定義されたＰＦキーが押されるとサブＣＰＵはメインＣＰＵに対してＦＩＲＱをかけます．

復帰情報　**エラーは生じません．**

解　説　割り込み用に設定されている場合には，ＰＦキーを押しても文字列は発生しません．

ＢＡＳＩＣの管理下であった場合には，＄５Ｂ１に割り込み処理のアドレスを書いておけば，ＢＡＳＩＣ内からそこへジャンプします（＄５Ｂ１に＄００００が書かれている場合にはジャンプしません．通常はこの形になっています）．この際Ｂレジスタには，ＰＦキー番号が入っています．しかしこの場合注意する必要があるのは，メインＣＰＵにはＦＩＲＱがかかりますが，ＢＡＳＩＣ内部で全レジスタを退避し，Ｅフラグを１にしているので，ユーザーは見かけ上ＩＲＱがかかったのと同じ状態で取り扱えるという点です（すなわち，Ｓレジスタをのぞいて破壊してもかまいません）．

サンプル　ＰＦキーを押すたびにメッセージを出力します．ＰＦ１０で終了します．

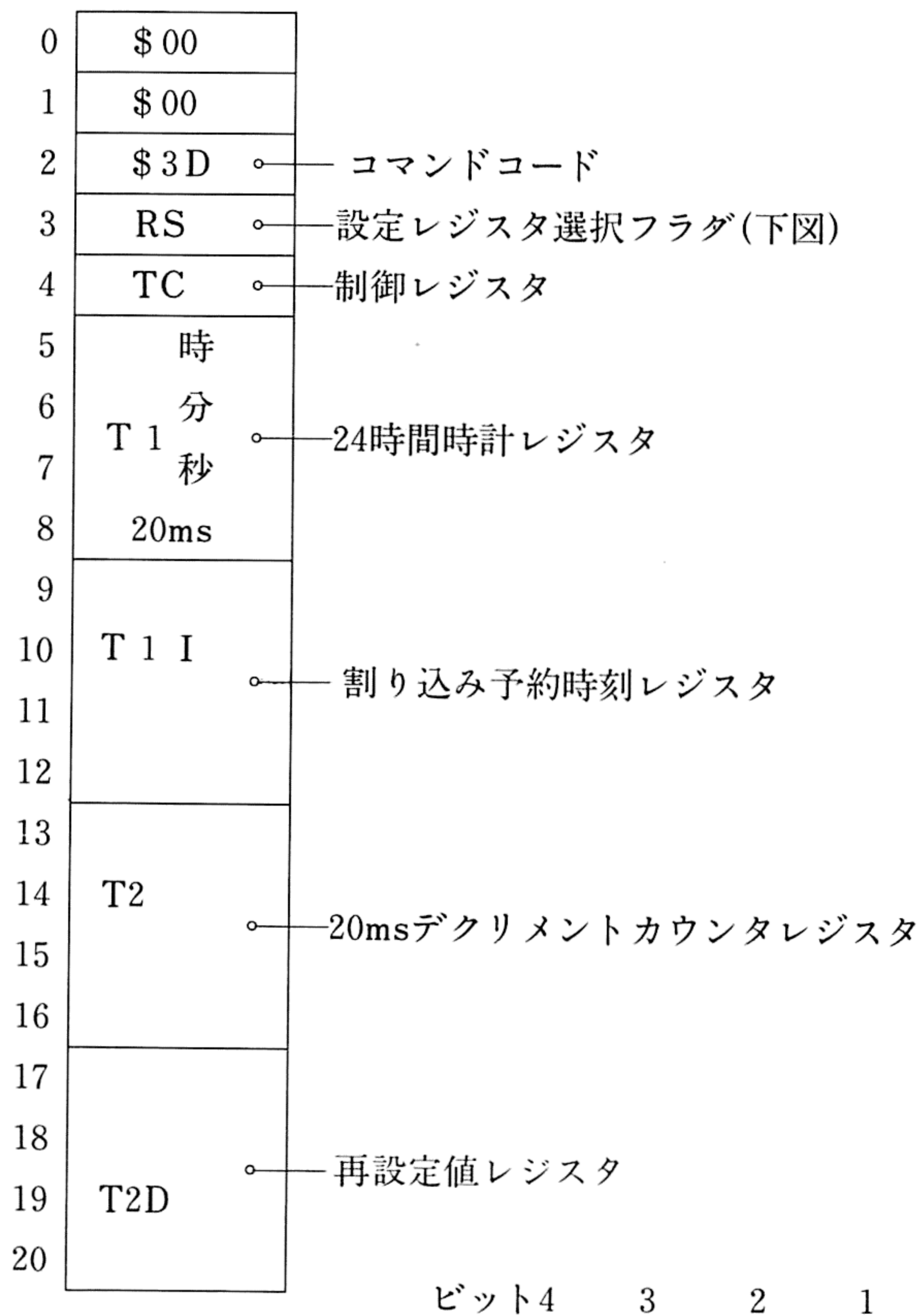
```
01000                         *
01010                         * PFkey interrupt sample
01020                         *
01030                                 OPT     M,NOS,NOG,NOO,PAGE=255
01040               9BDB      MESSAG  EQU     $9BDB    output from (X+1) till 0
01050               B615      D10OUT  EQU     $B615    output D reg in decimal
01060               FBFA      BIOS    EQU     $FBFA    bios (extend-indirect)
01070                         *
01080   5000                          ORG     $5000
01090               5000      START   EQU     *
01100                         *
01110                         * set interrupt PF1..PF10
01120                         *
01130   5000 CC     03FF              LDD     #%0000001111111111
01140   5003 8D     3D   5042         BSR     INTCTL
01150                         *
01160                         * vector set
01170                         *
01180   5005 1A     40                ORCC    #%01000000 FIRQ mask
01190   5007 30     8D 0010           LEAX    FIRQ,PCR set vector
01200   500B BF     05B1              STX     $5B1
01210   500E 1C     BF                ANDCC   #%10111111 FIRQ enable
01220   5010 6F     8D 0081           CLR     FLAG,PCR clear flag
01230   5014 6D     8D 007D   LOOP    TST     FLAG,PCR PF10 pressed ?
01240   5018 27     FA   5014         BEQ     LOOP     no, loop !!!
01250   501A 39                       RTS              yes,end of execution
01260                         *
01270                         * interrupt routine
01280                         *
01290   501B 34     04        FIRQ    PSHS    B        save PFkey no.
01300   501D 30     8D 003C           LEAX    MES-1,PCR output message
01310   5021 BD     9BDB              JSR     MESSAG
01320   5024 E6     E4                LDB     ,S       load PFkey_no.
01330   5026 4F                       CLRA
01340   5027 BD     B615              JSR     D10OUT   output PFkey_no.
01350   502A 35     04                PULS    B        PFkey_no. come back
01360   502C C1     0A                CMPB    #10      PF10 pressed ?
01370   502E 26     11   5041         BNE     FIRQ01   no.
01380                         *
01390                         * vector reset
01400                         *
01410   5030 1A     40                ORCC    #%01000000 FIRQ mask
01420   5032 CC     0000              LDD     #0       reset vector
01430   5035 FD     05B1              STD     $5B1
01440   5038 CC     0000              LDD     #%0000000000000000 PF1..PF10 reset
01450   503B 8D     05   5042         BSR     INTCTL
01460   503D 6C     8D 0054           INC     FLAG,PCR PF10 key flag on !
01470                         *
01480                         * return from interrupt
01490                         *
01500   5041 3B               FIRQ01  RTI
01510                         *
01520                         * subroutine:sybsys-command INTCTL execution
01530                         *
01540   5042 31     8C 57     INTCTL  LEAY    <SUBCOM,PCR
01550   5045 ED     23                STD     3,Y      set PFkey flags
01560   5047 30     8C 4C             LEAX    <RCB,PCR
01570   504A 86     10                LDA     #$10     bios request SUBOUT
01580   504C A7     84                STA     ,X
01590   504E 10AF   02                STY     2,X
01600   5051 CC     0005              LDD     #5       set length
01610   5054 ED     04                STD     4,X
01620   5056 86     2C                LDA     #$2C     subsys-command INTCTL
01630   5058 A7     22                STA     2,Y
01640   505A 6E     9F FBFA           JMP     [BIOS]   call bios & return
01650                         *
01660                         * message
01670                         *
01680   505E        0D0A      MES     FDB     $0D0A
01690   5060        50                FCC     'Programable function key was pressed !
!!  PFkey No.='
01700   5094        00                FCB     0
```

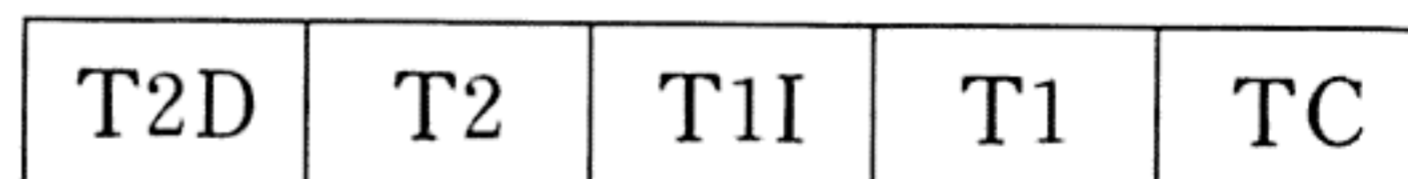
```
01710                         *
01720                         * working area
01730                         *
01740  5095      00           FLAG   FCB    0       PFkey pressed flag
01750  5096      0006         RCB    RMB    6
01760  509C      0005         SUBCOM RMB    5
01770                         *
01780            5000                END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50A0
PROGRAM ENTRY ADDR=5000


exec &H5000

Programable function key was pressed !!!  PFkey No.=1
Programable function key was pressed !!!  PFkey No.=3
Programable function key was pressed !!!  PFkey No.=5
Programable function key was pressed !!!  PFkey No.=7
Programable function key was pressed !!!  PFkey No.=9
Programable function key was pressed !!!  PFkey No.=10
Ready
```

SUB：SETIME　—タイマ設定—

機　能　タイマのレジスタに値を設定します.

パラメータ

0	$00	
1	$00	
2	$3D	コマンドコード
3	RS	設定レジスタ選択フラグ(下図)
4	TC	制御レジスタ
5	T1 時	24時間時計レジスタ
6	分	
7	秒	
8	20ms	
9–12	T1I	割り込み予約時刻レジスタ
13–16	T2	20msデクリメントカウンタレジスタ
17–20	T2D	再設定値レジスタ

RS：設定レジスタ選択フラグ

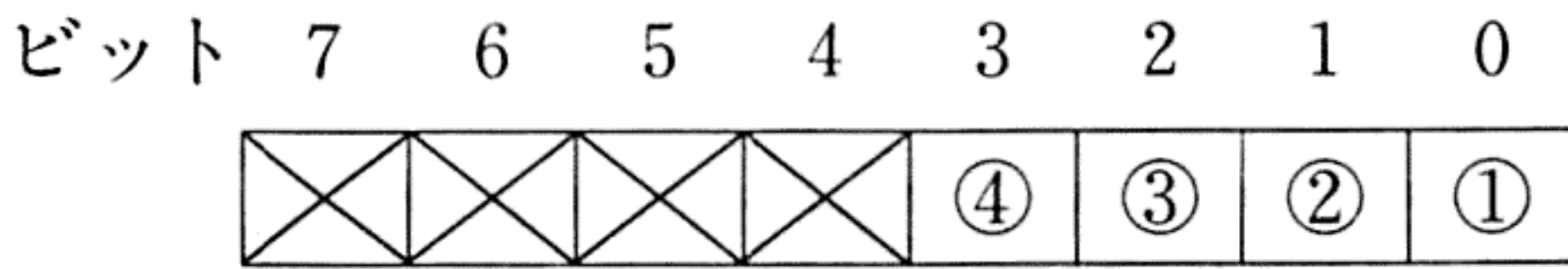
ビット4	3	2	1	0
T2D	T2	T1I	T1	TC

ビットが'1'のレジスタのみが，このコマンドによって設定されます.
'0'のレジスタは，現在の値が保持されます.

復帰情報　エラーは生じません.

解　説　設定レジスタとして選択しなかったレジスタの所は，パラメータとして与える必要はありません.

次に，各レジスタのレジスタの構成と役割を示します．

●ＴＣ：制御レジスタ

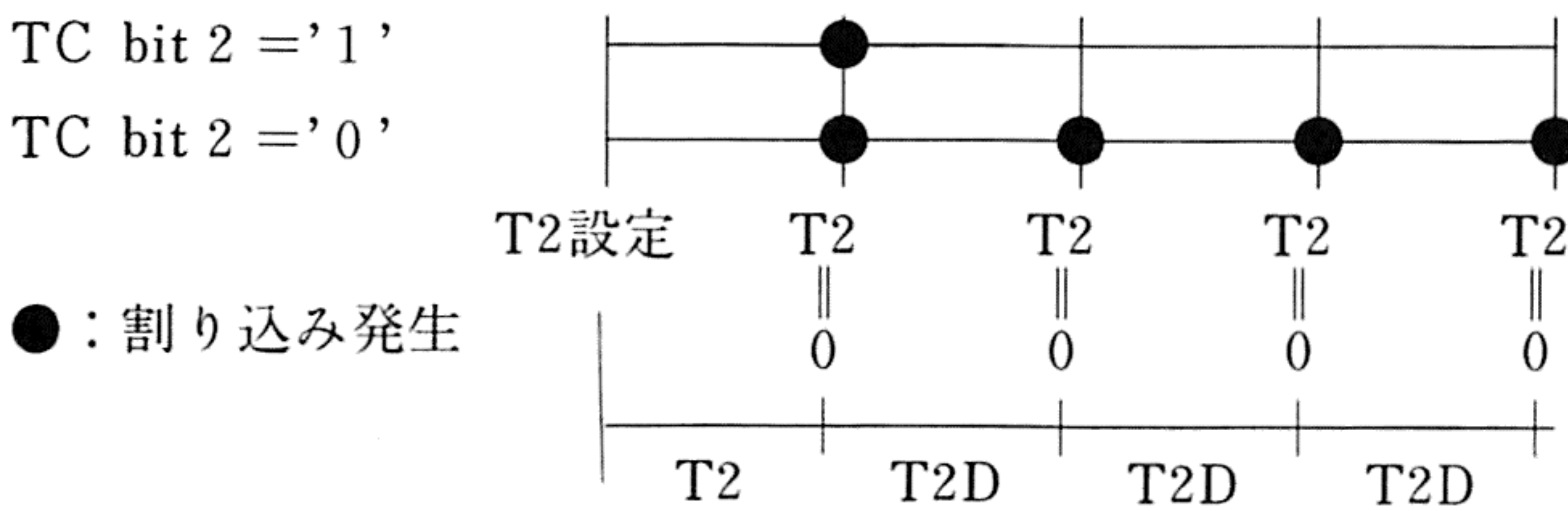
ビット	7	6	5	4	3	2	1	0
	╳	╳	╳	╳	④	③	②	①

①タイマ割り込みイネイブルフラグ　（1：イネイブル）

Ｔ１がカウントアップされた場合このフラグがＯＮで，かつ，Ｔ１とＴ１Ｉが等しければ，タイマ割り込みをメインＣＰＵにかけます（ＦＩＲＱ）．

②インターバルタイマ割り込みイネイブルフラグ　（1：イネイブル）

Ｔ２をカウントダウンして０となった場合に，このフラグがＯＮであればインターバルタイマ割り込みをメインＣＰＵにかけます（ＦＩＲＱ）．

③インターバルワンショットフラグ　（1：ワンショット）

Ｔ２をカウントダウンして０となった場合に，このフラグがＯＮであればＴＣのビット１を'0'にします．これは，インターバル割り込みを継続してかけるか，１回だけで停止にするかを設定するフラグで，このフラグにより下図のような違いが生じます（ただし，このフラグのＯＮ，ＯＦＦにかかわらず，Ｔ２が０となると，Ｔ２Ｄの内容をＴ２に設定します）

④０時割り込みイネイブルフラグ　（1：イネイブル）

Ｔ１をカウントアップされた時，このフラグがＯＮで，かつ，Ｔ１が０時を示していれば，０時割り込みをメインＣＰＵにかけます（ＦＩＲＱ）．

●Ｔ１：２４時間時計レジスタ

T1			
	0	時	(0～23)
	1	分	(0～59)
	2	秒	(0～59)
	3	20ms	(0～49)

このレジスタは，２０ｍｓごとにカウントアップします．秒，分，時へのくり上げは自動的に行なわれます．

カウントアップしてこのレジスタ

のすべての値が０（０時ちょうど）を示した時に，ＴＣレジスタのbit３（０時割り込みフラグ）がＯＮならば０時割り込みを発生します．

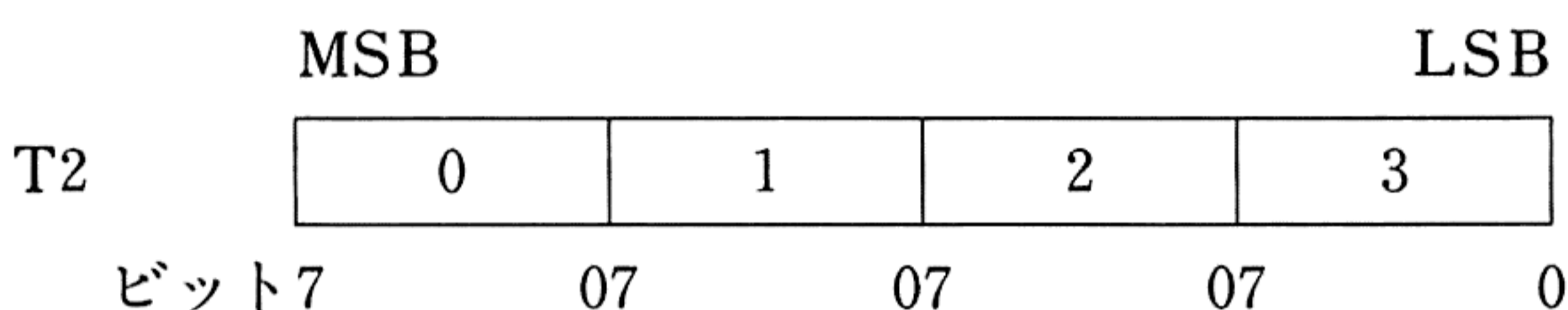
●Ｔ１Ｉ：割り込み予約時刻レジスタ

T1I			
0	時	(0～23)
1	分	(0～59)
2	秒	(0～59)
3	20ms	(0～49)

このレジスタは，タイマ割り込みを発生させる時刻を示すレジスタです．カウントアップ等は行なわれずＴ１と比較されるだけです．

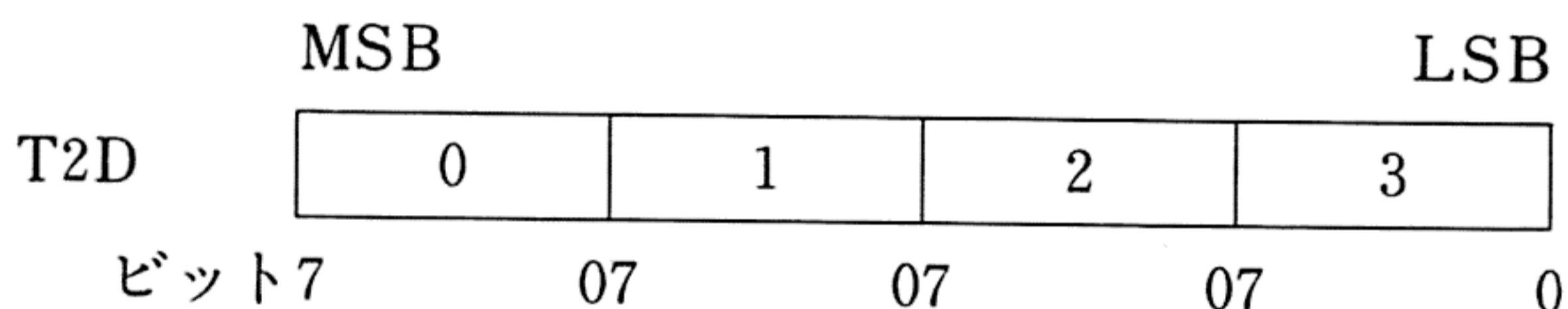
●Ｔ２：２０ｍｓデクリメントカウンタレジスタ（３２ビット）

MSB　LSB

T2	0	1	2	3
ビット	7　0	7　0	7　0	7　0

２０ｍｓごとにカウントダウンされます．カウントダウンして０となった時は，ＴＣのフラグによって割り込みを発生させます．また同時に，Ｔ２Ｄの値によって再設定され，カウントダウンを続行します．

●Ｔ２Ｄ：再設定値レジスタ（３２ビット）

MSB　LSB

T2D	0	1	2	3
ビット	7　0	7　0	7　0	7　0

このレジスタは，Ｔ２がカウントダウンされ０となったときに，Ｔ２に再設定される値を示します．このレジスタは，カウントダウン等は行なわれません．

これらのタイマによる割り込みはすべてＦＩＲＱで行なわれます．そのため，ユーザーはＦＩＲＱを生じた要因はなにかを，共有ＲＡＭのステータスを調べることにより識別する必要があります．(ＦＩＲＱはタイマ関係の他に，ＢＲＥＡＫキー，ＰＦキーによっても生じます)．またユーザーのルーチンで割り込みを処理したい場合には，ＦＩＲＱの割り込みベクトル（＄ＦＦＦ６，７）を書きかえる必要があります．

SUB：GETIME —タイマ読み取り—

機　能　タイマのレジスタをすべて読み取ります.

パラメータ

0	$00
1	$00
2	$3E — コマンドコード

復帰情報

0	$00
1	
2	
3	
4	TC — 制御レジスタ
5〜 8	T1 — 24時間時計レジスタ
9〜12	T1I — 割り込み予約時刻レジスタ
13〜16	T2 — 20msデクリメントレジスタ
17〜20	T2D — 再設定値レジスタ

各レジスタの構成についてはＳＥＴＩＭＥを御参照下さい.

サンプル

```
PAGE  001 (        ,        )

01000                             *
01010                             * SUBSYSTEM GETIME & SETIME
01020                             *
01030                                    OPT     M,NOS,P=255,NOO,NOG
01040                 FBFA        BIOS   EQU     $FBFA    bios (extend-indirect)
01050                 9BDB        MESSAG EQU     $9BDB    output from (X+1) till 0
01060                 D08E        OUT    EQU     $D08E    one character output
01070                 DB54        KEYIN  EQU     $DB54    keyinput with wait
01080                             *
01090   5000                             ORG     $5000
01100                 5000        START  EQU     *
01110                             *
01120                             * get timer
01130                             *
01140   5000 31       8D 00D3     GETIME LEAY    SUBCOM,PCR
01150   5004 86       3E                 LDA     #$3E     subsys-command GETIME
01160   5006 A7       22                 STA     2,Y      set command
01170   5008 30       8D 00C3            LEAX    RCB,PCR
01180   500C 86       11                 LDA     #$11     bios request SUBIN
01190   500E A7       84                 STA     ,X
01200   5010 10AF     02                 STY     2,X      set command addr.
```

```
01210  5013 CC    000F              LDD     #15       set command length
01220  5016 ED    04                STD     4,X
01230  5018 ED    06                STD     6,X
01240  501A AD    9F FBFA           JSR     [BIOS]    command execution
01250  501E 30    8D 0092           LEAX    MES1-1,PCR output message
01260  5022 BD    9BDB              JSR     MESSAG
01270  5025 C6    05                LDB     #5        counter
01280  5027 A6    A5          G01   LDA     B,Y
01290  5029 34    04                PSHS    B
01300  502B 5F                      CLRB              Breg=Areg/10
01310  502C 80    0A          G02   SUBA    #10       Areg=Areg MOD 10
01320  502E 25    03    5033        BCS     G03
01330  5030 5C                      INCB
01340  5031 20    F9    502C        BRA     G02
01350  5033 8B    0A          G03   ADDA    #10
01360  5035 34    02                PSHS    A         output in decimal
01370  5037 1F    98                TFR     B,A
01380  5039 8B    30                ADDA    #'0
01390  503B BD    D08E              JSR     OUT
01400  503E 35    02                PULS    A
01410  5040 8B    30                ADDA    #'0
01420  5042 BD    D08E              JSR     OUT
01430  5045 35    04                PULS    B
01440  5047 5C                      INCB              inc counter
01450  5048 C1    09                CMPB    #9        end ?
01460  504A 27    07    5053        BEQ     SETIME    yes.
01470  504C 86    3A                LDA     #':       no,output ':'
01480  504E BD    D08E              JSR     OUT
01490  5051 20    D4    5027        BRA     G01
01500                         *
01510                         * set timer
01520                         *
01530  5053 30    8D 006A     SETIME LEAX   MES2-1,PCR output message
01540  5057 BD    9BDB              JSR     MESSAG
01550  505A C6    05                LDB     #5        counter
01560  505C 34    04          S01   PSHS    B
01570  505E BD    DB54              JSR     KEYIN     decimal num. keyin
01580  5061 81    30                CMPA    #'0       test '0'..'9'
01590  5063 25    30    5095        BCS     S02
01600  5065 81    39                CMPA    #'9
01610  5067 22    2C    5095        BHI     S02
01620  5069 BD    D08E              JSR     OUT       echoback
01630  506C 80    30                SUBA    #'0
01640  506E C6    0A                LDB     #10       Areg=Areg*10
01650  5070 3D                      MUL
01660  5071 34    04                PSHS    B
01670  5073 BD    DB54              JSR     KEYIN     decimal num. keyin
01680  5076 81    30                CMPA    #'0       test '0'..'9'
01690  5078 25    1D    5097        BCS     S03
01700  507A 81    39                CMPA    #'9
01710  507C 22    19    5097        BHI     S03
01720  507E BD    D08E              JSR     OUT       echoback
01730  5081 80    30                SUBA    #'0
01740  5083 AB    E0                ADDA    ,S+       get number 0..99 in Areg
01750  5085 35    04                PULS    B
01760  5087 A7    A5                STA     B,Y       set number
01770  5089 5C                      INCB              inc counter
01780  508A C1    09                CMPB    #9        end ?
01790  508C 27    0B    5099        BEQ     S04       yes
01800  508E 86    3A                LDA     #':       no,output ':'
01810  5090 BD    D08E              JSR     OUT
01820  5093 20    C7    505C        BRA     S01
01830  5095 35    84          S02   PULS    B,PC      end of execution
01840  5097 35    86          S03   PULS    A,B,PC
01850                         *
01860                         * execute SETIME
01870                         *
01880  5099 CC    3D02        S04   LDD     #$3D02    subsys-com. SETIME
01890  509C ED    22                STD     2,Y       & set only 24hour timer
01900  509E 30    8D 002D           LEAX    RCB,PCR
01910  50A2 10AF  02                STY     2,X
01920  50A5 86    10                LDA     #$10      bios SUBOUT
01930  50A7 A7    84                STA     ,X
01940  50A9 CC    000F              LDD     #15       set command length
```

```
01950  50AC ED    04                  STD     4,X
01960  50AE AD    9F FBFA             JSR     [BIOS]  command execution
01970  50B2 16    FF4B 5000           LBRA    GETIME
01980                        *
01990                        * messages
02000                        *
02010  50B5       0D0A       MES1     FDB     $0D0A
02020  50B7       20                  FCC     '     TIME ='
02030  50C1       00                  FCB     0
02040  50C2       0D0A       MES2     FDB     $0D0A
02050  50C4       4E                  FCC     'NEW TIME ='
02060  50CE       00                  FCB     0
02070                        *
02080                        * working area
02090                        *
02100  50CF       0008       RCB      RMB     8
02110  50D7       000F       SUBCOM RMB      15
02120                        *
02130             5000                END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50E5
PROGRAM ENTRY ADDR=5000


time$="12:34:56"

Ready
exec &H5000

    TIME =12:35:05:16
NEW TIME =12:00:00:00
    TIME =12:00:00:00
NEW TIME =
Ready
? time$
12:00:08

Ready
```

SUB：CONT　—継続データ入出力—

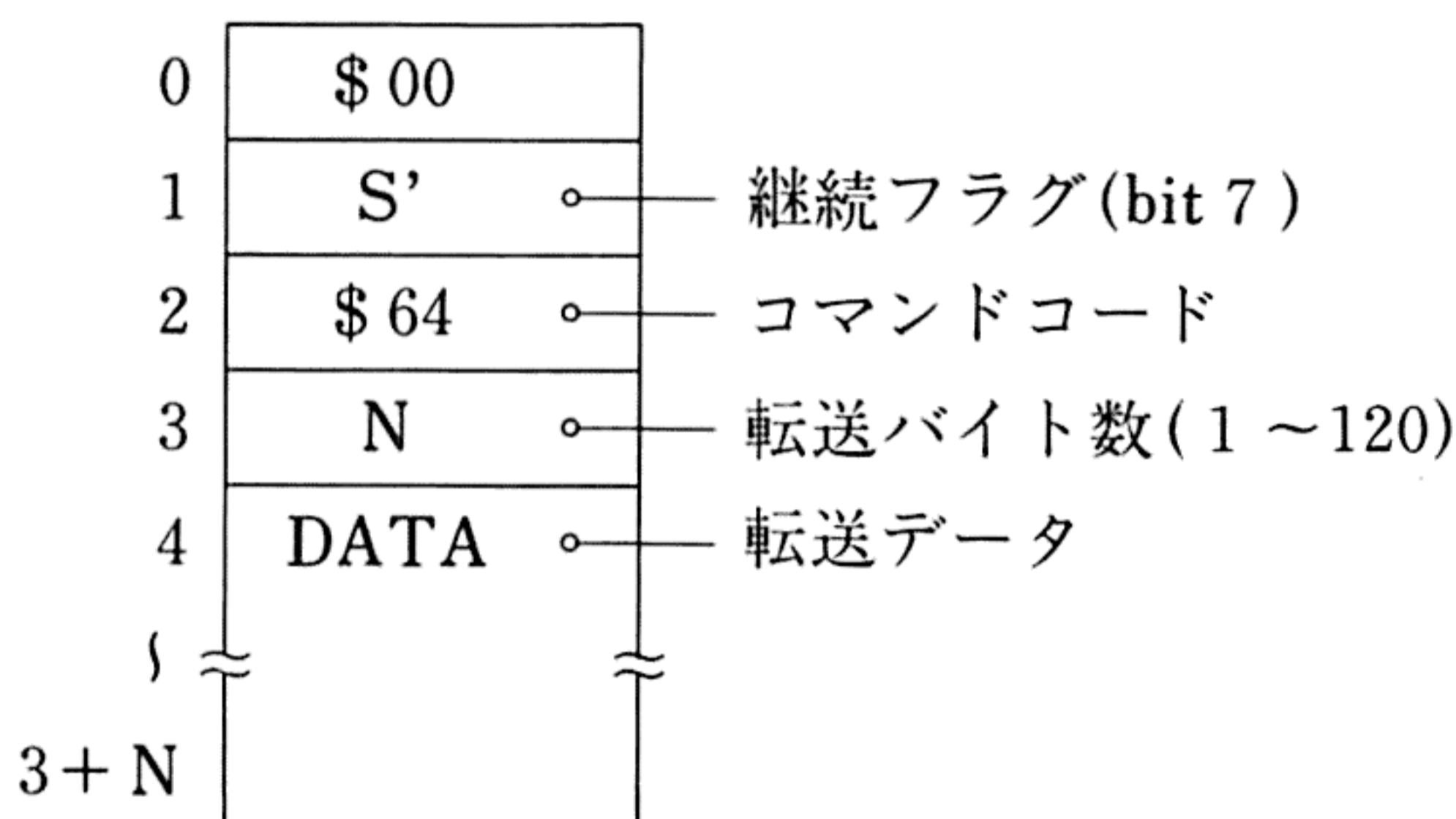
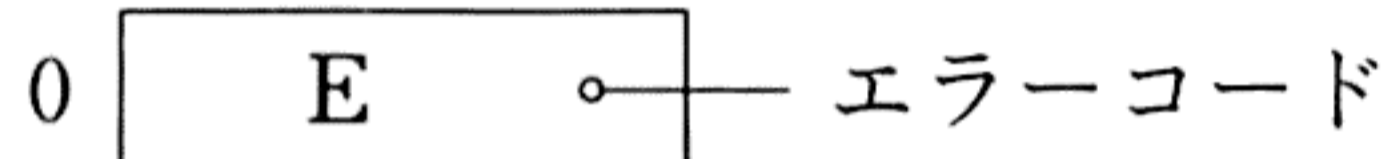
◀出力の場合▶

機　能　**サブシステムへのコマンド出力のときに，一度に転送しきれない分を転送します．**

パラメータ

0	$00	
1	S'	継続フラグ(bit 7)
2	$64	コマンドコード
3	N	転送バイト数(1～120)
4	DATA	転送データ
～		
3＋N		

復帰情報

0	E	エラーコード

解　説　サブシステムへコマンドを送出する場合，共有ＲＡＭが１２８バイトしかないことにより，コマンドのデータを一度に送出できない場合があります．この場合，最初にコマンドを送出するときに継続フラグを'1'にしてコマンドを送出して，サブシステムにまだコマンドが終了しないことを通知します．その後にこのコマンドで，残りのデータを送出します．さらに，未転送分が残る場合はこのコマンドを送る際に，継続フラグを'1'にすることにより続けてこのコマンドを送出してデータを送る必要があります．

このコマンドを使用する可能性のあるコマンドは，ＰＵＴ，ＧＥＴ，ＰＣＢＬＫ２，ＰＧＢＬＫ１，ＰＧＢＬＫ２の６つです．

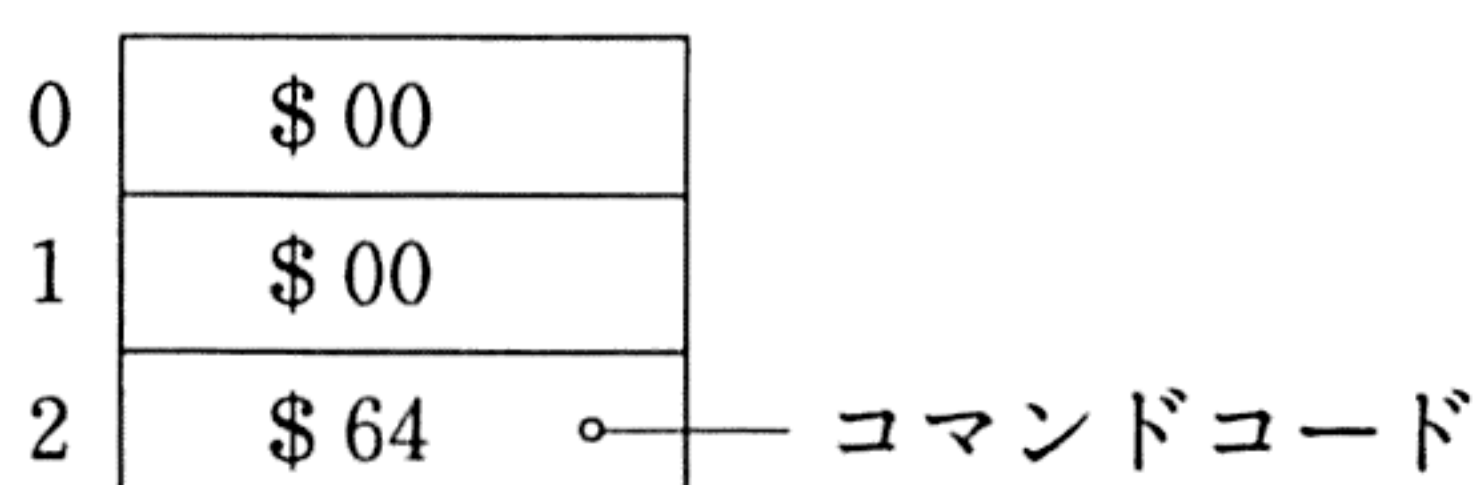
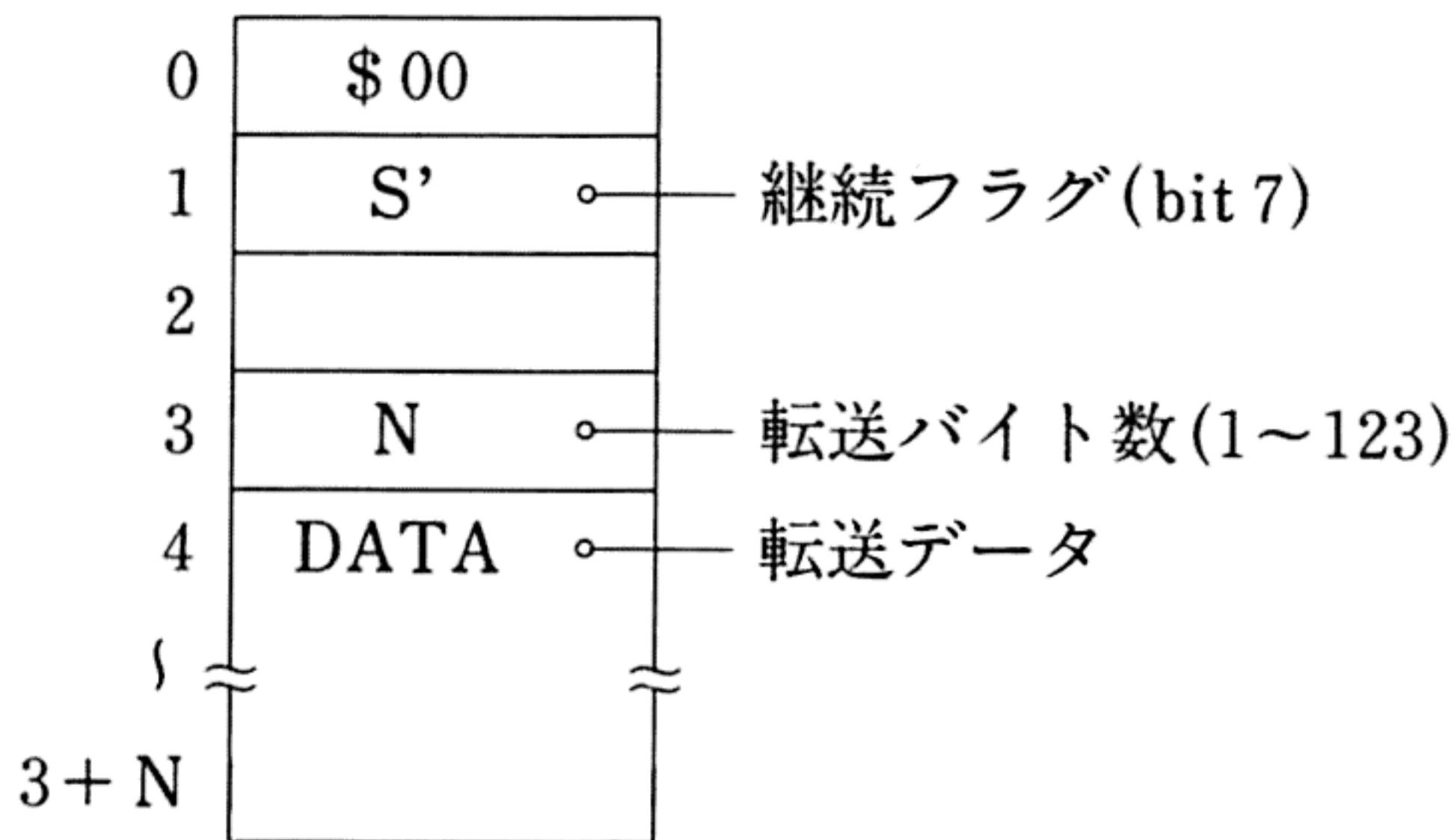
◀入力の場合▶

機　能　**サブシステムから返されるデータが一度に転送されなかった時，残りのデータの転送を要求します．**

パラメータ

0	$00	
1	$00	
2	$64	コマンドコード

復帰情報

0	$00	
1	S'	継続フラグ(bit 7)
2		
3	N	転送バイト数(1～123)
4	DATA	転送データ
≀		
3＋N		

解　　説 サブシステムからデータが返される時に，一度に転送しきれない場合サブシステムは継続フラグを'1'にしてデータを返します．継続フラグが'1'の時は，まだデータが続くことを意味しているので，このコマンドで残りのデータを要求し，受け取ります．このときさらに継続フラグが'1'の場合は，まだデータが残っているということなので，継続フラグが'0'になるまでこの動作をくり返す必要があります．

このコマンドを使用する可能性のあるコマンドは，ＧＥＴ，ＧＥＴＣ，ＧＣＢＬＫ１，ＧＣＢＬＫ２，ＧＧＢＬＫ１，ＧＧＢＬＫ２の６つです．

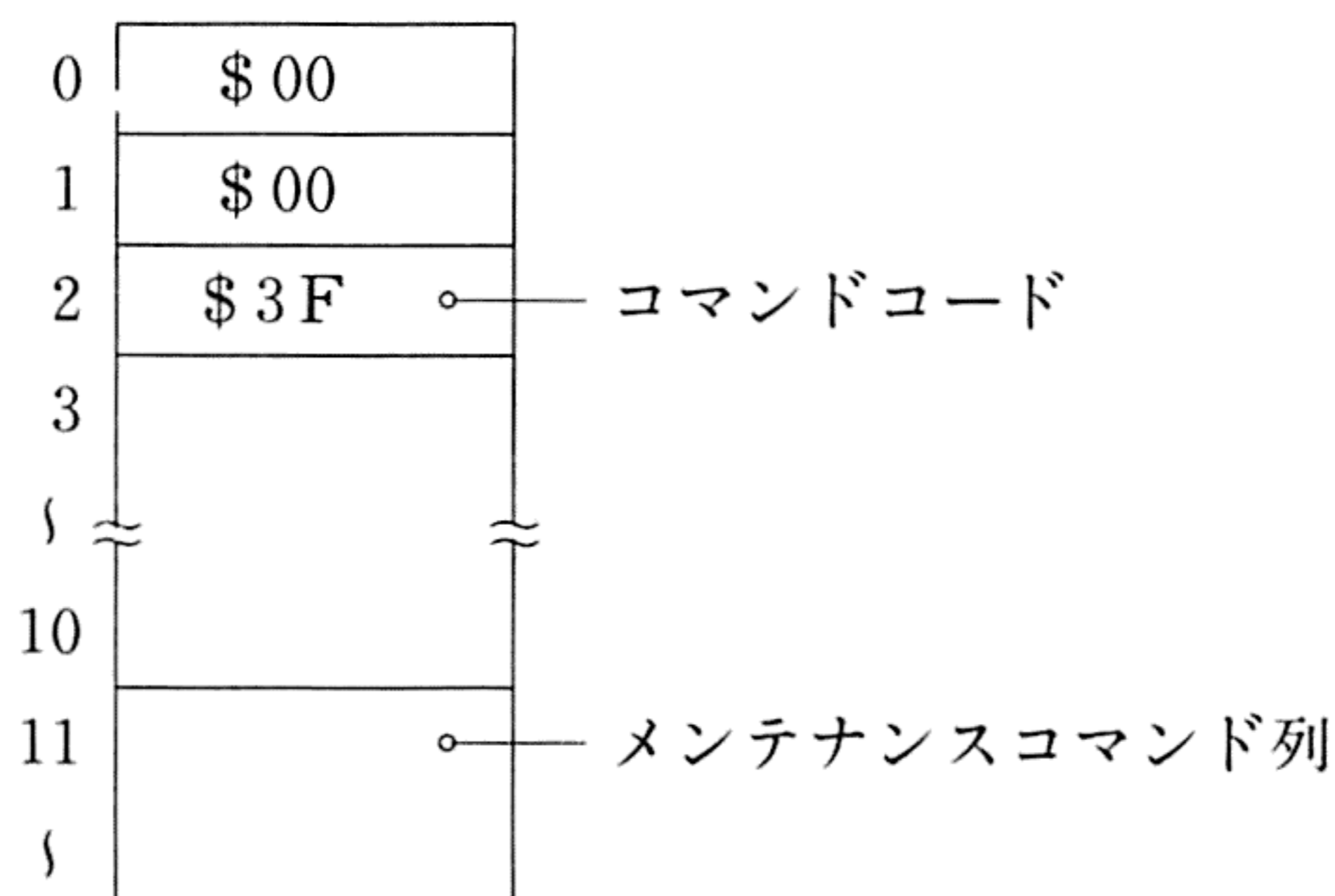
注）「ユーザーズマニュアルシステム仕様」，２－７１ページの復帰情報の項の相対値に訂正箇所があります．

誤	⇨	正
２，３	⇨	２
４	⇨	３
５～(４＋N)		４～(３＋N)

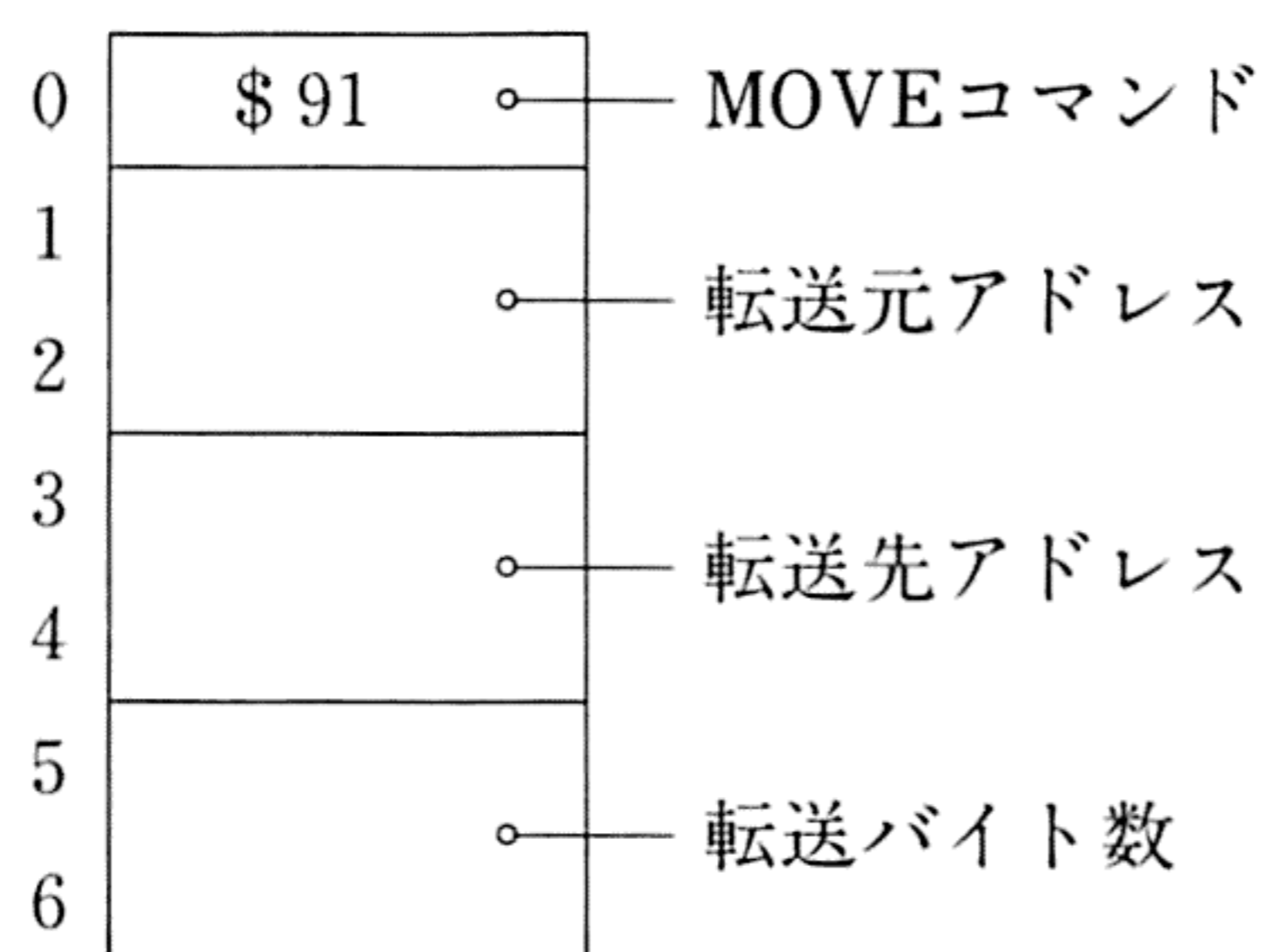
ＳＵＢ：ＴＥＳＴ　ーメンテナンス・コマンドー

機　能　サブＣＰＵのメモリにデータを転送したり，サブＣＰＵに独自のプログラムを実行させるためのコマンドです．

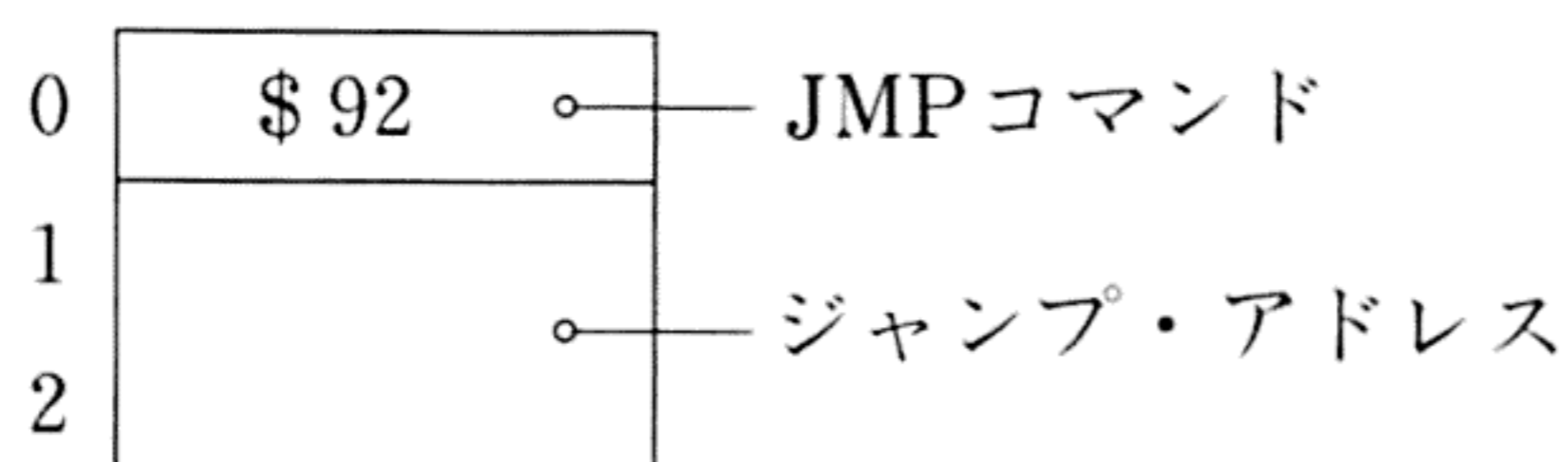
パラメータ

0	$00	
1	$00	
2	$3F	コマンドコード
3		
∫		
10		
11		メンテナンスコマンド列
∫		

◀メンテナンスコマンド列のフォーマット▶

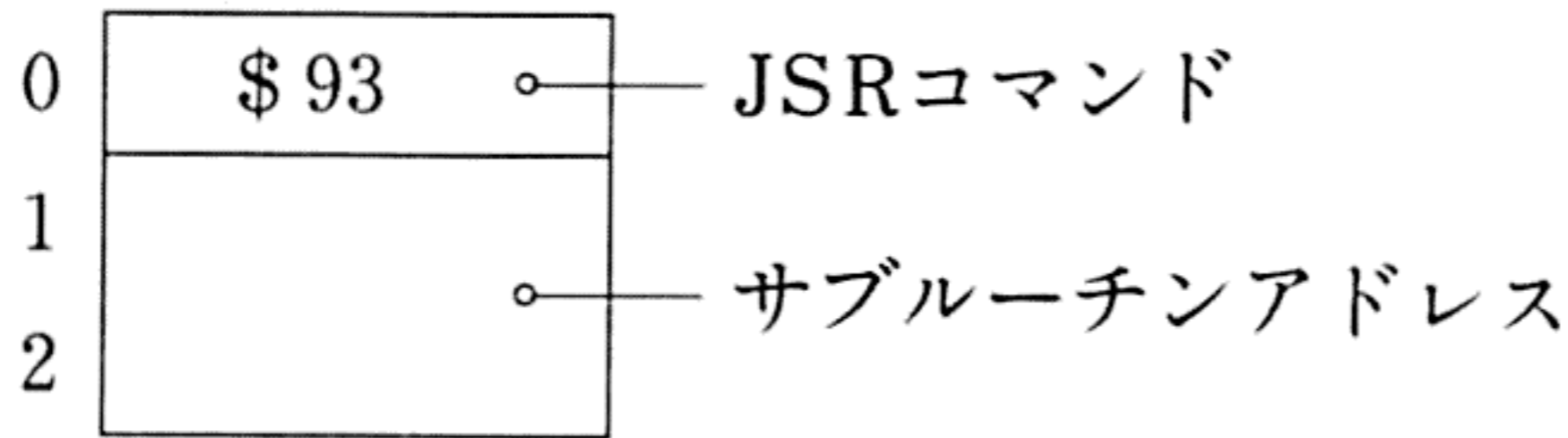
①ＭＯＶＥコマンド

0	$91	MOVEコマンド
1–2		転送元アドレス
3–4		転送先アドレス
5–6		転送バイト数

このコマンドはサブＣＰＵのメモリ内で転送を行なうものです．転送元，転送先のアドレスはサブＣＰＵのアドレスで指定して下さい．転送バイト数を０とすると６５５３６バイトで転送します．

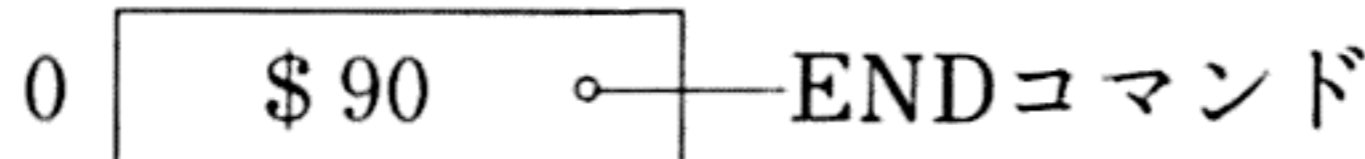
②ＪＭＰコマンド

0	$92	JMPコマンド
1–2		ジャンプ・アドレス

このコマンドはメンテナンスコマンドの制御を指定した番地にうつすものです．この場合も番地はサブＣＰＵの番地を指定して下さい．

③ＪＳＲコマンド

0	$93	JSRコマンド
1		サブルーチンアドレス
2		

このコマンドは，サブＣＰＵのメモリ上に存在するマシン語プログラム（ＲＯＭ上のもの，あるいは，ＲＡＭ上に転送したもの）をサブＣＰＵで実行するコマンドです．実行させるプログラムはＲＴＳ（または同等の命令）で終了するものに限ります．

④ＥＮＤコマンド

0	$90	ENDコマンド

このコマンドは，メンテナンスコマンド列の終了を指示するものです．メンテナンスコマンド列の最後に付加する必要があります．

解　説　このコマンドはこれらのサブコマンドを用いて，サブＣＰＵのメモリを直接参照するものです．これは「ユーザーズマニュアルシステム仕様」には示されていません．

ＦＭ８ではこのコマンドを使用する際，相対値３～１０にパスワード（'ＹＡＭＡＵＣＨＩ'）を設定する必要がありましたが，ＦＭ７ではパスワードの照合は行われないので設定する必要はありません．

サンプル　ごく簡単なゲームです．マシン語サブルーチンは，このコマンドを使用してダウンスクロールを行っています．

```
PAGE   001 (         .          )

01000                             *
01010                             * SUBSYSTEM TEST
01020                             *
01030                                     OPT     M,NOO,NOS,NOG,P=255
01040                 FBFA        BIOS    EQU     $FBFA    bios (extend-indirect)
01060                             *
01070   5000                              ORG     $5000
01080                 5000        START   EQU     *
01300                             *
01310                             * down scroll
01320                             *
01330   5000 31       8D 0089     DOWN    LEAY    SUBCOM,PCR load command addr.
01340   5004 30       8D 00AD             LEAX    RCB,PCR load rcb addr.
01350   5008 86       11                  LDA     #$11     bios SUBIN
01360   500A A7       84                  STA     ,X
```

```
01370  500C 10AF 02              STY    2,X      set command addr.
01380  500F CC   0028            LDD    #.SUBC-SUBCOM set command length
01390  5012 ED   04              STD    4,X
01400  5014 CC   0000            LDD    #0       return 0 bytes
01410  5017 ED   06              STD    6,x
01420  5019 AD   9F FBFA         JSR    [BIOS]   command execution
01430  501D 31   8D 0015         LEAY   MESD,PCR load command addr.
01440  5021 30   8D 0090         LEAX   RCB,PCR load rcb addr.
01450  5025 86   10              LDA    #$10     bios SUBOUT
01460  5027 A7   84              STA    ,X
01470  5029 10AF 02              STY    2,X      set command addr.
01480  502C CC   0057            LDD    #.MESD-MESD set command length
01490  502F ED   04              STD    4,X
01500  5031 AD   9F FBFA         JSR    [BIOS]   command execution
01510  5035 39                   RTS
01520                    *
01530                    * subsys-commands
01540                    *
01580  5036      00      MESD    FCB    0,0,$03,.MESD-*-1 command PUT
01590  503A      12              FCB    $12,0,0 locate 0,0:? spc$(80)
01600  503D      20              FCC    '
01610            508D    .MESD   EQU    *
01620                    *
01630                    * TEST command
01640                    *
01650            508D    SUBCOM  EQU    *
01660  508D      00              FCB    0,0,$3F command TEST
01670                    * pass word FM8='YAMAUCHI' , FM7= any string (len=8)
01680  5090      2A              FCC    '********'
01690  5098      93              FCB    $93      subcommand JSR
01700  5099      D38F            FDB    $D380+PROG-SUBCOM address of subcpu
01710  509B      90              FCB    $90      subcommand END
01720                    *
01730                    * scroll down (executed by subcpu)
01740                    *
01750            509C    PROG    EQU    *
01760  509C B7   D40A            STA    $D40A    BUSY flag on
01770  509F FC   D01F            LDD    $D01F    load vram offset
01780  50A2 B3   D03B            SUBD   $D03B    subtruction bytes of 1 line
01790  50A5 7D   D409            TST    $D409    vram access flag on
01800  50A8 FD   D01F            STD    $D01F    set vram offset(work)
01810  50AB FD   D40E            STD    $D40E    set vram offset(I/O)
01820  50AE B7   D409            STA    $D409    vram access flag off
01830  50B1 7D   D40A            TST    $D40A    BUSY flag off
01840  50B4 39                   RTS             return from subroutine
01850                    *
01860            50B5    .SUBC   EQU    *
01870                    *
01880                    * working area
01890                    *
01900  50B5      0006    RCB     RMB    6
01910                    *
01920            5000            END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50BA
PROGRAM ENTRY ADDR=5000
```

```
100 ' GAME PROGRAM for FM-7
110 ' (use subsystem TEST sample)
120 '           presented by H.Nakamura
130 PLAY "v10"
140 WIDTH 80,25
150 DIM X1(24),X2(24),XX(24)
160 SHIPLEFT=3:STAGE=1:SCORE=0
170 SHIP=40:DIR=0:X1=30+(STAGE MOD 7):X2=50-(STAGE MOD 7)
180 CLS:COLOR7
190 FOR P=0 TO 24:X1(P)=-1:X2(P)=80:XX(P)=-1:NEXT:P=0
200 LOCATE 0,0:PRINT STRING$(X1+1,"×");TAB(X2);STRING$(79-X2,"×");
210 FOR I=0 TO 200
220 COLOR 6
```

```
230 XX=-1:IF I MOD 15=0 THEN XX=X1+INT(RND*(X2-X1-1))+1:LOCATE XX,0:PRINT"♥";
240 COLOR 7
250 LOCATE X1,0:PRINT"×";:LOCATE X2,0:PRINT"×"
260 X1(P)=X1:X2(P)=X2:XX(P)=XX:P=P+1
270 IF P =25 THEN P=0
280 COLOR 5
290 LOCATE SHIP,24:PRINT"♠";
300 A$=INKEY$
310 IF A$="4" THEN DIR=-1:GOTO340
320 IF A$="6" THEN DIR=1:GOTO340
330 IF A$<>"" THEN DIR=0
340 SHIP=SHIP+DIR
350 IF SHIP<0 THEN SHIP=0
360 IF SHIP>79 THEN SHIP=79
370 PP=P+1:IF PP=25 THEN PP=0
380 IF SHIP<=X1(PP) OR SHIP>=X2(PP) THEN 480
390 IF SHIP=XX(PP) THEN PLAY"o6l16ceg":SCORE=SCORE+10
400 EXEC &H5000:' down scroll
410 S=INT(RND*3)-1
420 X1=X1+S:IF X1<0 THEN X1=0:GOTO 440
430 X2=X2+S:IF X2>79 THEN X2=79:X1=X1-S
440 NEXT
450 COLOR4:LOCATE 25,10:PRINT"**** CLEAR STAGE";STAGE;"****"
460 PLAY"o6l16cegcegcegcegceg":SCORE=SCORE+50:STAGE=STAGE+1
470 GOTO 530
480 COLOR2:LOCATE 25,10:PRINT"****  C L A S H ! ! ! ****"
490 SOUND 7,&H35:SOUND 8,&H10
500 SOUND 6,&H1F:SOUND 12,8
510 SOUND 13,0
520 SHIPLEFT=SHIPLEFT-1
530 COLOR5:LOCATE 25,12:PRINT"  score =";SCORE
540 IF SHIPLEFT=0 THEN 580
550 LOCATE 25,14:PRINT"  ship left :";SHIPLEFT
560 FOR I=0 TO 5000:NEXT
570 GOTO 170
580 COLOR2:LOCATE 25,16:PRINT"**** G A M E  O V E R ****"
590 FOR I=0 TO 2000:NEXT
600 FOR I=50 TO 30STEP -1:PLAY"l32n=I;":NEXT
610 FOR I=0 TO 2000:NEXT
620 PRINT CHR$(27)+CHR$(&H39)
630 COLOR7
640 LOCATE 0,0:END
```

3. System Subroutines 1

本章は，F－BASIC Ver 3.0のROM上にある各種のルーチンのうち，数値演算関係以外のルーチンで，一般ユーザーの利用頻度が高いと思われるものを選び出し，解説を加えたものです．

◀各項目のみかた▶

各ルーチンは用途別に分類してあります．

アドレス 各ルーチンのエントリポイントを示します．

レジスタ 各ルーチンの実行後，内容が変化する可能性のあるレジスタ名を略号で示します．したがって，各ルーチンを使用するにあたって保存したいレジスタが含まれている場合には，ユーザーの側でレジスタの退避が必要です．

機　能，**解　説**，**サンプル**，**パラメータ**，**復帰情報** についてはBIOSの章と同様です．ただし，パラメータ，復帰情報は必要のある場合だけ記述します．

なお，パラメータの値で特記しませんが，この章のルーチンをコールするに当って，DPレジスタは0である必要があります．

＄ＦＥ００：システムリセット

機　能　リセットボタンが押されたのと同様の状況となります．

アドレス　＄ＦＥ００

解　説　ＦＭ７のリセットボタンが押されると，ＣＰＵはこの番地へジャンプしてきますので，ユーザーがここへジャンプすると，リセットキーが押されたのと同様の状況になります．ただし，このときＢＲＥＡＫキーが押された状態であればＢＡＳＩＣがホットスタート（Abort）します．

このルーチンの処理をフローチャートに示します．(次頁)

＄８４８Ｂ：ＢＡＳＩＣコールドスタート

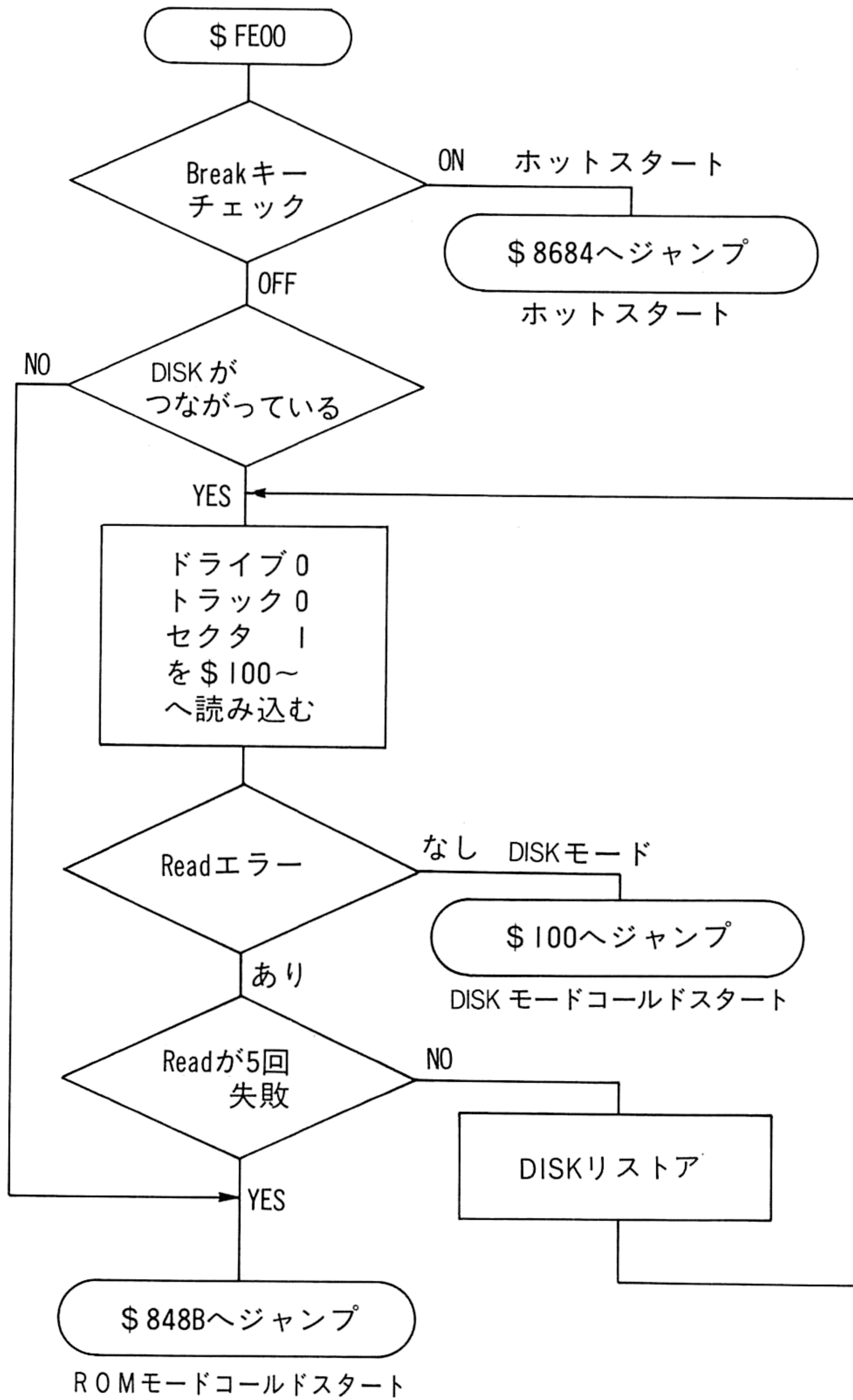
機　能　ＢＡＳＩＣをコールドスタートします．

パラメータ　Ａレジスタ←モード　ＲＯＭモード　＝０
　　　　　　　　　　　　　　　　ＤＩＳＫモード≠０
　　　　　　Ｘレジスタ←イニシャライズルーチンのエントリアドレス
　　　　　　　　　　　（ＤＩＳＫモードを指定した場合のみ）

アドレス　＄８４８Ｂ

解　説　ＢＡＳＩＣを指定したモードでコールドスタートします．DISKモードを要求する場合には，あらかじめＤＯＳがＲＡＭ上にロードされている必要があります．

このルーチンは，ワークエリア，Ｉ／Ｏの初期化，Free Areaの０クリア等を行い，ＢＡＳＩＣのコマンドレベルへ制御を移します．

リセットルーチンの処理

＄８６８４：ＢＡＳＩＣホットスタート

機　能　ＢＡＳＩＣのホットスタートを行います．この際，ＲＡＭ上のプログラムは保存されます．

アドレス　＄８６８４

解　説　ＢＡＳＩＣのホットスタートを行います．ＲＡＭ上のプログラムはクリアされませんが，画面表示は初期化されます（ＢＡＳＩＣの変数も保存されます）．

このルーチンでは，コールに当ってＤＰレジスタの値は０であることを必要としません．

またこのルーチンは，ユーザーによってウォームスタート(「ユーザーズマニュアルシステム解説」10.3参照）がかけられたのと同様の動作をし，Ｆ－ＢＡＳＩＣのコマンドレベルにもどります．

サンプル

```
PAGE  001 (          ,          )

01000                         *
01010                         * basic hot start
01020                         *              ($8684)
01030                              OPT    M,NOO,NOS,NOG,P=255
01040             8684        ABORT  EQU    $8684   basic hot start
01050             9BDB        MESSAG EQU    $9BDB
01060             DB54        KEYIN  EQU    $DB54   keyinput
01070    5000                        ORG    $5000
01080             5000        START  EQU    *
01090                         *
01100    5000 30  8D 0026            LEAX   MES-1,PCR
01110    5004 BD  9BDB               JSR    MESSAG
01120    5007 BD  DB54        LOOP   JSR    KEYIN
01130    500A 81  72                 CMPA   #'r
01140    500C 27  15   5023          BEQ    RET
01150    500E 81  68                 CMPA   #'h
01160    5010 26  F5   5007          BNE    LOOP
01170                         *
01180                         * basic hot start (Abort)
01190                         *
01200    5012 30  8D 003A            LEAX   MESHOT-1,PCR
01210    5016 BD  9BDB               JSR    MESSAG
01220    5019 8E  FFFF               LDX    #$FFFF
01250    501C 30  1F          WAIT   LEAX   -1,X    wait a moment
01260    501E 26  FC   501C          BNE    WAIT
01270    5020 7E  8684               JMP    ABORT
01280                         *
01290                         * return to basic
01300                         *
01310    5023 30  8D 003C     RET    LEAX   MESRET-1,PCR
01320    5027 BD  9BDB               JSR    MESSAG
01330    502A 39                     RTS
01340                         *
01350    502B     0D0A        MES    FDB    $0D0A
01360    502D     20                 FCC    ' Basic hot start or return (h/r) ? '
01370    5050     00                 FCB    0
01380    5051     42          MESHOT FCC    'Basic hot start !!'
```

```
01390  5063      00              FCB    0
01400  5064      72       MESRET FCC    'return to basic !!'
01410  5076      00              FCB    0
01420                     *
01430            5000            END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5076
PROGRAM ENTRY ADDR=5000


exec &H5000

 Basic hot start or return (h/r) ? return to basic !!
Ready
exec

 Basic hot start or return (h/r) ? Basic hot start !!




Abort
Ready
```

$D678：ABORTテスト

機　能　BREAKキーが押されたかをチェックし，押されていた場合には，'Abort' を出力後，BASICのコマンドレベルへもどります．

アドレス　$D678

レジスタ　BREAKキーが押されていなかった場合にはCCのみ破壊し，BREAKキーが押されていた場合にはもどりません．

解　説　BREAKキーが押されたかどうかをチェックし，押されていた場合には "Abort" のメッセージを出力した後，BASICのコマンドレベルへ制御を移します．

ここで注意しなければならないのは，このABORTテストは$313の内容を参照して実行されている点です．これは，BREAKキーが押されると，メインCPUにFIRQがかかり，$313に$40が代入されることを利用したものです．よって，このルーチンが呼ばれた時に，BREAKキーをチェックするわけではありません．

以上のことは，BACICプログラム実行中のBREAKキー入力にも当てはまります．すなわち，ユーザーがBREAKキーを押しても，その時すぐにBREAKがかかるのではなく$313に$40が代入されるだけで，BREAKがかかるのはBASIC内部で$313の内容を調べた時になります．

また，BREAKキーの処理は，FIRQを利用しているため，ユーザーによって，又は，BASIC内部でFIRQがマスクされている場合には，BREAKキーが押されても，無視されます．

サンプル

```
PAGE  001 (        ,        )

01000                           *
01010                           * basic abort test
01020                           *
01030                                   OPT     M,NOS,NOG,NOO,PAGE=255
01040               9BDB        MESSAG  EQU     $9BDB   output from (X+1) till 0
01050               D678        ABORTT  EQU     $D678   abort test
01080                           *
01090  5000                             ORG     $5000
01100               5000        START   EQU     *
01110                           *
01120                           * output message & loop until Break
01130                           *
01140  5000 30      8D 0007             LEAX    MES-1,PCR
01150  5004 BD      9BDB                JSR     MESSAG
01160  5007 BD      D678        LOOP    JSR     ABORTT  ** abort test
```

```
01170  500A 20   FB   5007          BRA     LOOP
01180                          *
01190                          * message
01200                          *
01210  500C      0D0A         MES       FDB     $0D0A
01220  500E      70                     FCC     'push BREAK key !!!
01230  5021      00                     FCB     0
01240                          *
01250            5000                   END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5021
PROGRAM ENTRY ADDR=5000


exec &H5000

push BREAK key !!!

Abort
Ready
```

$８ＦＥ１：ＢＲＥＡＫエントリ

機　能　実行中のＦ－ＢＡＳＩＣプログラムを停止して，コマンドレベルへもどします．

パラメータ　ＣＣレジスタ・キャリーフラグ（bit 0）

←｛ 'Break' のメッセージを出力する　：1
　　'Break' のメッセージを出力しない：0

アドレス　$８ＦＥ１

解　説　現在実行中のＦ－ＢＡＳＩＣプログラムの実行を中止して，ＢＡＳＩＣのコマンドレベルへ制御を移します．マシン語のサブルーチンなどから直接ＢＡＳＩＣのコマンドレベルへ戻りたい場合などに利用します．このルーチンを呼び出す際のキャリーフラグのＯＮ，ＯＦＦにより，"Break" のメッセージを出力するか，しないかを選択することができます．

具体的には以下のことを実行します．

- ハードウェアスタックを補正します．（このルーチンへのエントリは，ＪＳＲ，ＪＭＰのどちらでも可能です）．
- $４９，$４Ａに現在実行中だった行の行番号をいれる．
- $４Ｄ，$４Ｅをセットする．（実行中のステートメントのあるアドレス）
- "Break" のメッセージを出力します．（エントリ時のキャリーフラグによる）．
- ＢＡＳＩＣのコマンドレベルへ制御を移す．

サンプル

```
PAGE  001 (        ,        )

01000                             *
01010                             * Break entry
01020                             *                    ($8FE1)
01030                                     OPT     M,NOO,NOS,NOG,P=255
01040                 8FE1        BREAK   EQU     $8FE1    basic Break
01050                 9BDB        MESSAG  EQU     $9BDB    output from (X+1) till 0
01060                 DB54        KEYIN   EQU     $DB54    keyinput
01070       5000                          ORG     $5000
01080                 5000        START   EQU     *
01090                             *
01100       5000 30   8D 0025             LEAX    MES-1,PCR
01110       5004 BD   9BDB                JSR     MESSAG
01120       5007 BD   DB54        LOOP    JSR     KEYIN
01130       500A 81   79                  CMPA    #'y
01140       500C 27   10   501E           BEQ     YES
01150       500E 81   6E                  CMPA    #'n
01160       5010 26   F5   5007           BNE     LOOP
01170                             *
01180                             * Break without 'Break'
```

```
01190                          *
01200  5012 30   8D 0048             LEAX   MESNO-1,PCR
01210  5016 BD   9BDB                JSR    MESSAG
01220  5019 1C   FE                  ANDCC  #$FE     clear carry
01230  501B 7E   8FE1                JMP    BREAK
01240                          *
01250                          * Break with 'Break'
01260                          *
01270  501E 30   8D 002C       YES   LEAX   MESYES-1,PCR
01280  5022 BD   9BDB                JSR    MESSAG
01290  5025 1A   01                  ORCC   #$01     set carry
01300  5027 7E   8FE1                JMP    BREAK
01310                          *
01320  502A      0D0A          MES   FDB    $0D0A
01330  502C      20                  FCC    ' Basic BREAK with message (y/n) ? '
01340  504E      00                  FCB    0
01350  504F      77            MESYES FCC   'with message !!'
01360  505E      00                  FCB    0
01370  505F      77            MESNO  FCC   'without message !!'
01380  5071      00                  FCB    0
01390                          *
01400            5000                END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5071
PROGRAM ENTRY ADDR=5000


exec &H5000

 Basic BREAK with message (y/n) ? without message !!
Ready
exec

 Basic BREAK with message (y/n) ? with message !!

Break
Ready
```

＄８ＤＤ１：エラー処理

機　能　ＢＡＳＩＣのエラーメッセージを出力して，ＢＡＳＩＣのコマンドレベルへ制御を移します．

パラメータ　Ｂレジスタ←エラーコード（「Ｆ－ＢＡＳＩＣ文法書」付録５参照）

アドレス　＄８ＤＤ１

解　説　エラーコードで指定するエラーメッセージを出力し，ＢＡＳＩＣのコマンドレベルへ制御を移します．

エラーコードは0以外の数を指定して下さい．エラーメッセージが定義されていないコードの場合は 'Unprintable Error' のメッセージを出力します．

また，ＤＩＳＫモードでない時にディスク関係のエラーコード（６５～７３）を指定した場合も 'Unprintable Error' となります．

サンプル

```
PAGE  001 (                )

01000                              *
01010                              * error message output
01020                              *                ($8DD1)
01030                                   OPT     M,NOS,NOG,NOO,PAGE=255
01040                 9BDB         MESSAG EQU   $9BDB    output from (X+1) till 0
01050                 D08E         OUT    EQU   $D08E    one character output
01060                 8DD1         ERROR  EQU   $8DD1    basic error
01070                 DB54         KEYIN  EQU   $DB54    keyinput with wait
01080                              *
01090  5000                             ORG     $5000
01100                 5000         START  EQU   *
01110                              *
01120                              * get error code in Breg
01130                              *
01140  5000 30       8D 0024              LEAX    MES-1,PCR output message
01150  5004 BD       9BDB                 JSR     MESSAG
01160  5007 4F                            CLRA             get number in Breg
01170  5008 5F                            CLRB
01180  5009 34       02           LOOP   PSHS    A
01190  500B 86       0A                   LDA     #10      Breg=Breg*10+keyin
01200  500D 3D                            MUL
01210  500E EB       E0                   ADDB    ,S+
01220  5010 34       04                   PSHS    B
01230  5012 BD       DB54                 JSR     KEYIN    keyinput
01240  5015 35       04                   PULS    B
01250  5017 81       30                   CMPA    #'0      test '0'..'9'
01260  5019 25       0B    5026           BCS     ERR
01270  501B 81       39                   CMPA    #'9
01280  501D 22       07    5026           BHI     ERR
01290  501F BD       D08E                 JSR     OUT      echo back
01300  5022 80       30                   SUBA    #'0
01310  5024 20       E3    5009           BRA     LOOP
01320                              *
01330                              * jump to error routine
01340                              *
01350  5026 7E       8DD1          ERR    JMP     ERROR
01360                              *
01370                              * message
01380                              *
```

```
01390  5029      0D0A      MES   FDB   $0D0A
01400  502B      42              FCC   'BASIC error code = '
01410  503E      00              FCB   0
01420            5000            END   START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=503E
PROGRAM ENTRY ADDR=5000
```

```
exec &H5000

BASIC error code = 2

Syntax Error
Ready
exec

BASIC error code = 4

Out Of Data
Ready
exec

BASIC error code = 100

Unprintable Error
Readv
```

＄８Ｆ３９：プログラム・クリア

機　能　ＢＡＳＩＣプログラムを消去（ＮＥＷ）します.

アドレス　＄８Ｆ３９（ＢＡＳＩＣプログラムの消去と同時にオープンされている，ファイルのクローズを行いたい場合は＄８Ｆ３６）

解　説　ＢＡＳＩＣプログラムを消去（ＮＥＷ）します．具体的には，ＢＡＳＩＣのテキストのリンクポインタをイニシャライズすることにより消去します.（＄３３のさすアドレスとその次のアドレスに0を書きこみ，＄３５に，＄３３の内容に2を加えた値を代入する）

またこのルーチンは，内部でＳレジスタ（スタックポインタ）を変更しているので，これにはＪＭＰで入る必要があります.（スタックのトップにあるアドレスだけが保存されます）

サンプル

```
PAGE  001 (        .         )

01000                       *
01010                       * BASIC program clear
01020                       *               ($8F39)
01030                            OPT    M,NOO,NOS,NOG,P=255
01040             8F39      CLEAR  EQU    $8F39   basic program clear
01050             9BDB      MESSAG EQU    $9BDB   output from (X+1) till 0
01060             DB54      KEYIN  EQU    $DB54   keyinput
01070  5000                        ORG    $5000
01080             5000      START  EQU    *
01090                       *
01100  5000 30    8D 001F          LEAX   MES-1,PCR
01110  5004 BD    9BDB             JSR    MESSAG
01120  5007 BD    DB54      LOOP   JSR    KEYIN
01130  500A 81    79               CMPA   #'y
01140  500C 27    0C    501A       BEQ    YES
01150  500E 81    6E               CMPA   #'n
01160  5010 26    F5    5007       BNE    LOOP
01170                       *
01180                       * not clear program & return to basic
01190                       *
01200  5012 30    8D 0047          LEAX   MESNO-1,PCR
01210  5016 BD    9BDB             JSR    MESSAG
01220  5019 39                     RTS
01230                       *
01240                       * clear program & jump to basic
01250                       *
01260  501A 30    8D 0023   YES    LEAX   MESYES-1,PCR
01270  501E BD    9BDB             JSR    MESSAG
01280  5021 7E    8F39             JMP    CLEAR
01290                       *
01300  5024       0D0A      MES    FDB    $0D0A
01310  5026       63               FCC    'clear text program (y/n) ? '
01320  5041       00               FCB    0
01330  5042       79        MESYES FCC    'yes.
01340  5046       0D0A             FDB    $0D0A
01350  5048       63               FCC    'clear text program !!'
01360  505D       00               FCB    0
01370  505E       6E        MESNO  FCC    'no.
01380  5061       0D0A             FDB    $0D0A
```

```
01390  5063      6E                FCC    'not clear text program !!'.
01400  507C      00                FCB    0
01410                      *
01420            5000              END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=507C
PROGRAM ENTRY ADDR=5000


LIST

10 ' program text
20 '
30 A$=" program here !!!!"
40 PRINT A$

Ready
RUN
 program here !!!!

Ready
exec &H5000

clear text program (y/n) ? no.
not clear text program !!
Ready

LIST


10 ' program text
20 '
30 A$=" program here !!!!"
40 PRINT A$

Ready
RUN
 program here !!!!

Ready
exec

clear text program (y/n) ? yes.
clear text program !!
Ready
LIST


Ready
RUN

Ready
```

$8F1E：行サーチ

機　能　テキスト中から特定の行を探し出します.

パラメータ　Dレジスタ←行番号

アドレス　$8F1E

レジスタ　CC, X, U

復帰情報　CCレジスタのキャリーフラグ(C), ゼロフラグの内容によって結果が分かれます.

(1)　C＝0, Z＝1の場合

指定した行番号の行が存在しています. Xレジスタに指定行の先頭アドレス(リンク・ポインタを含みます)が, Uレジスタには次の行の先頭アドレスがセットされます.

(2)　C＝1, Z＝0の場合

指定した行番号の行は存在しませんが, 指定した行番号より大きな行番号をもつ行が存在します. このとき, Xレジスタにその行の先頭アドレス, Uレジスタにその次の行の先頭アドレスがセットされます.

(3)　C＝1, Z＝1の場合

指定した行番号の行は存在せず, かつ, それより大きい行番号の行も存在しないことを示します. Xレジスタには, F－BASICのテキストエンドを表わす, 2バイトの＄００が格納されている最初のアドレスを指します. またUレジスタには0がセットされます.

サンプル

```
PAGE  001 (        ,         )

01000                           *
01010                           * line search
01020                           *              ($8F1E)
01030                                  OPT     M,NOS,NOG,NOO,PAGE=255
01040             9BDB          MESSAG EQU     $9BDB    output from (X+1) till 0
01050             D08E          OUT    EQU     $D08E    one character output
01060             8F1E          SEARCH EQU     $8F1E    line search
01070             DB54          KEYIN  EQU     $DB54    keyinput with wait
01080             AC37          D16OUT EQU     $AC37    output Dreg in HEX
01090                           *
01100  5000                            ORG     $5000
01110             5000          START  EQU     *
01120                           *
01130                           * get line no. in Dreg
01140                           *
01150  5000 30    8D 004A              LEAX    MES-1,PCR output message
01160  5004 BD    9BDB                 JSR     MESSAG
01170  5007 CC    0000                 LDD     #0
01180  500A 34    06                   PSHS    D        number
```

```
01190  500C 34    06          GL01   PSHS   D          keyinput
01200  500E EC    62                 LDD    2,S        get number
01210  5010 58                       ASLB              Dreg=Dreg*10
01220  5011 49                       ROLA
01230  5012 58                       ASLB
01240  5013 49                       ROLA
01250  5014 E3    62                 ADDD   2,S
01260  5016 58                       ASLB
01270  5017 49                       ROLA
01280  5018 E3    E1                 ADDD   ,S++       add keyinput
01290  501A ED    E4                 STD    ,S         store number
01300  501C BD    DB54               JSR    KEYIN      get keyinput
01310  501F 81    30                 CMPA   #'0        test '0'..'9'
01320  5021 25    0E    5031         BCS    GL02
01330  5023 81    39                 CMPA   #'9
01340  5025 22    0A    5031         BHI    GL02
01350  5027 BD    D08E               JSR    OUT
01360  502A 80    30                 SUBA   #'0
01370  502C 1F    89                 TFR    A,B
01380  502E 4F                       CLRA
01390  502F 20    DB    500C         BRA    GL01
01400  5031 35    06          GL02   PULS   D          number in Dreg
01410  5033 BD    8F1E               JSR    SEARCH     ** line search
01420  5036 25    0F    5047         BCS    NOFIND     if not found
01430  5038 34    10                 PSHS   X          found
01440  503A 30    8D 001E            LEAX   FINDM-1,PCR
01450  503E BD    9BDB               JSR    MESSAG
01460  5041 35    06                 PULS   D
01470  5043 BD    AC37               JSR    D16OUT     output address
01480  5046 39                       RTS
01490                         *
01500  5047 30    8D 0020     NOFIND LEAX   NOFINM-1,PCR output message
01510  504B BD    9BDB               JSR    MESSAG
01520  504E 39                       RTS
01530                         *
01540                         * messages
01550                         *
01560  504F       0D0A        MES    FDB    $0D0A
01570  5051       20                 FCC    ' LINE NO. ='
01580  505C       00                 FCB    0
01590  505D       0D0A        FINDM  FDB    $0D0A
01600  505F       20                 FCC    ' LINE FROM $'
01610  506B       00                 FCB    0
01620  506C       0D0A        NOFINM FDB    $0D0A
01630  506E       20                 FCC    ' LINE NOT FOUND !! '
01640  5081       00                 FCB    0
01650                         *
01660             5000               END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5081
PROGRAM ENTRY ADDR=5000


LIST

10 ' line 10
20 ' line 20
30 ' last line of program

Ready
exec &H5000

 LINE NO. =10
 LINE FROM $0C6A
Ready
exec

 LINE NO. =30
 LINE FROM $0C88
Ready
exec
```

```
 LINE NO. =40
 LINE NOT FOUND !!
Ready
```

```
BEG ADDR 0C6A
END ADDR 0CA4
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F        CHARACTER
0C60   00 00 00 00 00 00 00 00 00 00 0C 79 00 0A 3A 8D     ...........y..:■
0C70   20 6C 69 6E 65 20 31 30 00 0C 88 00 14 3A 8D 20      line 10..|..:■
0C80   6C 69 6E 65 20 32 30 00 0C A4 00 1E 3A 8D 20 6C     line 20.....:■ l
0C90   61 73 74 20 6C 69 6E 65 20 6F 66 20 70 72 6F 67     ast line of prog
0CA0   72 61 6D 00 00 00 00 00 00 00 00 00 00 00 00 00     ram.............
```

$C730：リンク・ポインタ設定

機　能　BASICテキストのリンクポインタを再設定します.

パラメータ　($33, $34)←BASICテキストの先頭番地

アドレス　$C730

レジスタ　CC, A, B, X, U

復帰情報　Xレジスタ：プログラムエンドを示す2バイトの$00が格納されている最初のアドレスを指します.

解　説　$33, $34に格納されているプログラムの先頭アドレスからXレジスタをポインタとして，プログラムエンドへ向けてサーチして行き，次の行の先頭を見つけたら，現在のリンクポインタに次の行の先頭アドレスを代入します. この作業を次行の先頭2バイト（リンクポインタ）が$00になるまで続けます.

サンプル　NEWされたプログラムを復活させます（リセットによって失われたプログラムは復活不可能です）.

```
PAGE  001 (        ,        )

01000                       *
01010                       * BASIC TEXT recovery program
01020                       *                ($C730)
01030                              OPT    M,NOS,NOG,NOO,PAGE=255
01040              9BDB     MESSAG EQU    $9BDB    output from (X+1) till 0
01050              C730     LINK   EQU    $C730    link basic text
01060              AC37     D16OUT EQU    $AC37    output Dreg in HEX
01070                       *
01080  5000                        ORG    $5000
01090              5000     START  EQU    *
01100                       *
01110                       * output message & link BASIC text
01120                       *
01130  5000 30     8D 0022         LEAX   MES1-1,PCR
01140  5004 BD     9BDB            JSR    MESSAG
01150  5007 DC     33              LDD    $33      load text start address
01160  5009 BD     AC37            JSR    D16OUT
01170  500C 30     8D 0048         LEAX   MES2-1,PCR
01180  5010 BD     9BDB            JSR    MESSAG
01190                       *
01200                       * recover program main
01210                       *
01220  5013 CC     0001            LDD    #1       set first link pointer
01230  5016 ED     9F 0033         STD    [$33]    as not end of program
01240  501A BD     C730            JSR    LINK     ** link BASIC program
01250  501D 30     02              LEAX   2,X      set text end address
01260  501F 9F     35              STX    $35
01270                       *
01280  5021 1F     10              TFR    X,D
01290  5023 BD     AC37            JSR    D16OUT
01300  5026 39                     RTS
01310                       *
01320                       * messages
```

```
01330                           *
01340  5027      0D0A      MES1   FDB    $0D0A
01350  5029      2A               FCC    '** BASIC TEXT RECOVER **'
01360  5041      0D0A             FDB    $0D0A
01370  5043      54               FCC    'TEXT START ADDRESS =$'
01380  5058      00               FCB    0
01390  5059      0D0A      MES2   FDB    $0D0A
01400  505B      54               FCC    'TEXT END   ADDRESS =$'
01410  5070      00               FCB    0
01420                           *
01430            5000             END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5070
PROGRAM ENTRY ADDR=5000
```

```
LIST

10 ' program text
20 PRINT" program text here !!"

Ready
RUN
 program text here !!

Ready
new

Ready
LIST


Ready
RUN

Ready

exec &H5000

** BASIC TEXT RECOVER **
TEXT START ADDRESS =$0C6A
TEXT END   ADDRESS =$0C9D
Ready
LIST

10 ' program text
20 PRINT" program text here !!"

Ready
RUN
 program text here !!

Ready
```

＄C６３D：BASICプログラム実行

機　能　**メモリ上に格納されているBASICプログラムを実行します．**

アドレス　**＄C６３D**

解　説　このルーチンはBASICプログラムを実行しますが，トレースフラグ（＄A９）がONされている場合は，トレース動作も同時に行います．

このルーチンを使用するに当っては，BASICの各種のポインタがセットされている必要があります．これは，下記の「BASIC用ポインタ設定」を使用することによって行われますので，ユーザーは，このルーチンを使用する前に下記のルーチンを実行しておく必要があります．

＄８F４B：BASIC用ポインタ設定

機　能　**BASIC内で使用しているポインタ類を，BASICプログラム実行のためにセットします．**

アドレス　**＄８F４B：通常の場合**
＄８F４３：UNLIST，Protect の解除を行う場合
＄８F４１：＄８F４３の動作に加えて，トレースフラグのクリアをする場合

レジスタ　**CC，A，B，X，Y，U**

解　説　このルーチンは，＄C６３Dと組み合わせてBASICのRUNに相当する動作をしますが，RUNと異なりオープンされているファイルのクローズは行いません．

このルーチンは，内部でBASICテキストの読み出しポインタ（＄D９）を書きかえています．したがって，BASICからEXEC　&H８F４B，又はモニタからG８F４Bを行った場合には，それだけで，BASICプログラムを実行します．

サンプル

```
PAGE  001 (        ,        )

01000                       *
01010                       * execute BASIC program
01020                       *                    ($8F4B,$C63D)
01030                              OPT    M,NOS,NOG,NOO,PAGE=255
01040             9BDB      MESSAG EQU    $9BDB   output from (X+1) till 0
01050             8F4B      INIT   EQU    $8F4B   initialize for basic
01060             C63D      EXEC   EQU    $C63D   execute basic program
01070                       *
01080  5000                        ORG    $5000
01090             5000      START  EQU    *
01100                       *
01110                       * output message & start BASIC program
01120                       *
01130  5000 30   8D 0008          LEAX   MES-1,PCR
01140  5004 BD   9BDB             JSR    MESSAG
01150  5007 BD   8F4B             JSR    INIT
01160  500A 7E   C63D             JMP    EXEC
01170                       *
01180  500D      0D0A      MES    FDB    $0D0A
01190  500F      65               FCC    'execute basic program !!'
01200  5027      0D0A             FDB    $0D0A
01210  5029      00               FCB    0
01220                       *
01230             5000             END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5029
PROGRAM ENTRY ADDR=5000


LIST

10 ' program text
20 '
30 PRINT" program text here !!!"

Ready
mon

*G5000

execute basic program !!
 program text here !!!

Ready
```

＄ＡＢＦ４：モニタエントリ

機　　能　モニタのコマンドレベルへ制御を移します．

アドレス　＄ＡＢＦ４

解　　説　＄ＡＢＦ４にジャンプすることでモニタに制御が移ります．これを利用することにより，モニタをマシン語デバッガ（機能はごく最小限のものとなります）として使用することができます．

つまり，ユーザープログラム中にＪＭＰ＄ＡＢＦ４を書き込んでブレークポイントとしておくと，そこを実行した時にモニタに飛んでくるため，Ｒコマンドを使用して，その場所でのレジスタ値を調べることができます．

サンプル

```
PAGE  001 (         ,         )

01000                        *
01010                        * use MONITOR(MON) as debugger
01020                        *              ($ABF4)
01030                               OPT     M,NOS,NOG,NOO,PAGE=255
01040           9BDB         MESSAG EQU     $9BDB   output from (X+1) till 0
01050           D08E         OUT    EQU     $D08E   one character output
01060           DB54         KEYIN  EQU     $DB54   keyinput with wait
01070           AC37         D16OUT EQU     $AC37   output Dreg in HEX
01080           ABF4         MON    EQU     $ABF4   monitor entry
01090                        *
01100  5000                         ORG     $5000
01110           5000         START  EQU     *
01120                        *
01130                        * break point set   :START
01140                        * break point reset :START+2
01150                        *
01160  5000 20  02   5004           BRA     SET
01170  5002 20  6E   5072           BRA     RESET
01180                        *
01190                        * set break point
01200                        *      get break point address
01210                        *
01220  5004 30  8D 008A      SET    LEAX    MES1-1,PCR output message
01230  5008 BD  9BDB                JSR     MESSAG
01240  500B CC  0000                LDD     #0
01250  500E 34  06                  PSHS    D
01260  5010 34  06           GA01   PSHS    D        save keyin
01270  5012 EC  62                  LDD     2,S      Dreg=Dreg*16+keyin
01280  5014 58                      ASLB
01290  5015 49                      ROLA
01300  5016 58                      ASLB
01310  5017 49                      ROLA
01320  5018 58                      ASLB
01330  5019 49                      ROLA
01340  501A 58                      ASLB
01350  501B 49                      ROLA
01360  501C E3  E1                  ADDD    ,S++
01370  501E ED  E4                  STD     ,S
01380  5020 BD  DB54                JSR     KEYIN    keyinput
01390  5023 81  30                  CMPA    #'0      test '0'..'9' or 'A'..'F'
01400  5025 25  1D   5044           BCS     GA04
01410  5027 81  39                  CMPA    #'9
01420  5029 22  07   5032           BHI     GA02
```

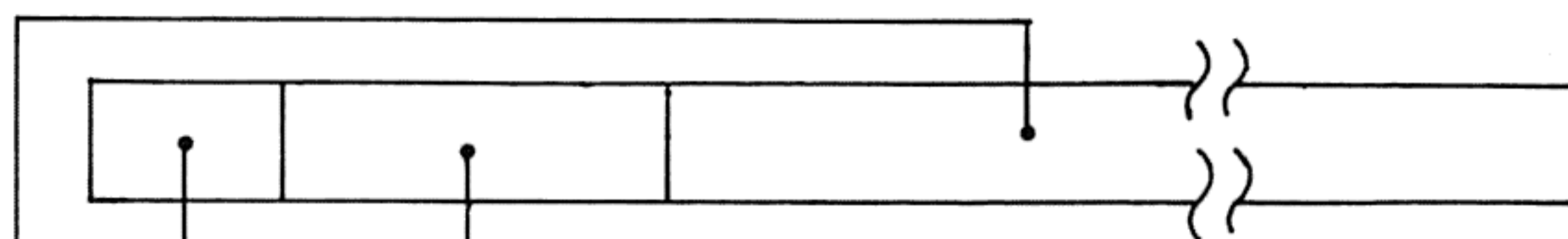
```
01430  502B BD   D08E               JSR    OUT
01440  502E 80   30                 SUBA   #'0
01450  5030 20   0D   503F          BRA    GA03
01460  5032 81   41          GA02   CMPA   #'A
01470  5034 25   0E   5044          BCS    GA04
01480  5036 81   46                 CMPA   #'F
01490  5038 22   0A   5044          BHI    GA04
01500  503A BD   D08E               JSR    OUT
01510  503D 80   37                 SUBA   #'A-10
01520  503F 1F   89          GA03   TFR    A,B
01530  5041 4F                      CLRA
01540  5042 20   CC   5010          BRA    GA01
01550  5044 35   06          GA04   PULS   D        address in Dreg
01560                        *
01570  5046 1F   01                 TFR    D,X
01580  5048 AF   8D 0090            STX    ADDR,PCR save address
01590  504C A6   84                 LDA    ,X       save memory
01600  504E A7   8D 008C            STA    MEM1,PCR
01610  5052 EC   01                 LDD    1,X
01620  5054 ED   8D 0087            STD    MEM2,PCR
01630  5058 86   7E                 LDA    #$7E     write JMP MON
01640  505A A7   84                 STA    ,X
01650  505C CC   ABF4               LDD    #MON
01660  505F ED   01                 STD    1,X
01670  5061 30   8D 003E            LEAX   MES2-1,PCR output message
01680  5065 BD   9BDB               JSR    MESSAG
01690  5068 EC   8D 0070            LDD    ADDR,PCR
01700  506C BD   AC37               JSR    D16OUT
01710  506F 7E   ABF4               JMP    MON      jump to monitor
01720                        *
01730                        * reset break point
01740                        *
01750  5072 30   8D 0044     RESET  LEAX   MES3-1,PCR output message
01760  5076 BD   9BDB               JSR    MESSAG
01770  5079 EC   8D 005F            LDD    ADDR,PCR load set address
01780  507D BD   AC37               JSR    D16OUT
01790  5080 A6   8D 005A            LDA    MEM1,PCR memory recover
01800  5084 AE   8D 0054            LDX    ADDR,PCR
01810  5088 A7   84                 STA    ,X
01820  508A EC   8D 0051            LDD    MEM2,PCR
01830  508E ED   01                 STD    1,X
01840  5090 7E   ABF4               JMP    MON      jump to monitor
01850                        *
01860  5093      0D0A        MES1   FDB    $0D0A
01870  5095      53                 FCC    'SET ADDRESS =$'
01880  50A3      00                 FCB    0
01890  50A4      0D0A        MES2   FDB    $0D0A
01900  50A6      42                 FCC    'BREAK POINT SET AT $'
01910  50BA      00                 FCB    0
01920  50BB      0D0A        MES3   FDB    $0D0A
01930  50BD      42                 FCC    'BREAK POINT RESET   ADDRESS =$'
01940  50DB      00                 FCB    0
01950                        *
01960                        *
01970                        *
01980  50DC      0000        ADDR   FDB    0        break point address
01990  50DE      12          MEM1   FCB    $12      memory save area
02000  50DF      1212        MEM2   FDB    $1212
02010                        *
02020            5000               END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50E0
PROGRAM ENTRY ADDR=5000


mon

*G5000

SET ADDRESS =$4026
```

```
BREAK POINT SET AT $4026
*G4000

BASIC error code = 255
*R
CC 89-
A  0D-
B  FF-
DP 00-
X  05A0-
Y  C4E5-
U  1340-
PC ABF9-

*


*G5002

BREAK POINT RESET   ADDRESS =$4026
*G4000

BASIC error code = 255

Unprintable Error
Ready
```

＄ＣＤＤ１：ＢＡＳＩＣプログラム・セーブ

機　能	ＢＡＳＩＣのプログラムをセーブします．
パラメータ	＄２Ｅ６←物理機番 ＄２ＤＤ←ファイル名長 ＄２ＤＥ～＄２Ｅ５←ファイル名 ＄ＢＦ←定数（＄１１） ＄２ＤＢ，＄２ＤＣ，＄２Ｄ４←０
アドレス	＄ＣＤＤ１
レジスタ	ＣＣ，Ａ，Ｂ，Ｘ（指定する機器によってＹ，Ｕ）

解　説　メモリ上にあるＢＡＳＩＣプログラムをセーブします．

ファイルのフォーマットは，下図のようになります．

UNLISTの行番号

このルーチンでは常に＄ＦＦ
（プロテクトをかけてセーブすると＄ＦＥとなる）

メモリ上に展開されているコードがそのまま出力される．
（プロテクトがかかっている場合には特殊な処理が施してある）

＄ＣＥ１Ｂ：マシン語プログラム・セーブ

機　能　メモリ上のマシン語プログラムをセーブします．

パラメータ　＄２Ｅ６←物理機番

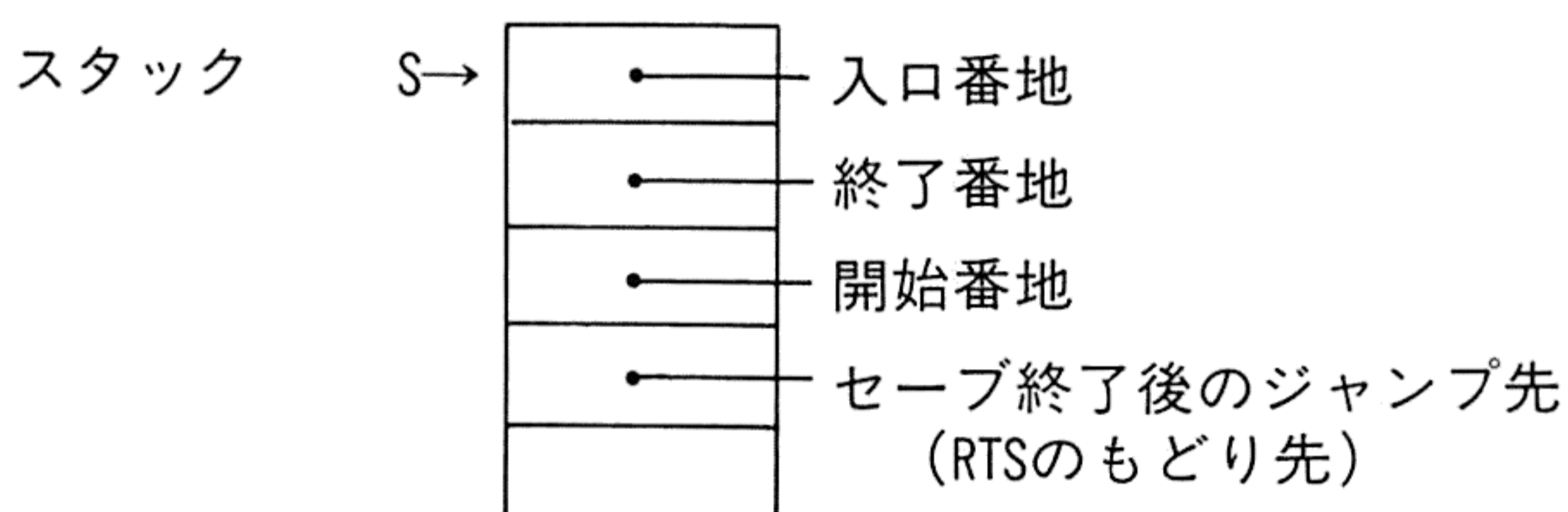
＄２ＤＤ←ファイル名長

＄２ＤＥ～＄２Ｅ５←ファイル名

＄ＢＦ←定数（＄１１）

＄２ＤＣ，＄２Ｄ４←０

スタック　S→

入口番地
終了番地
開始番地
セーブ終了後のジャンプ先（RTSのもどり先）

アドレス　＄ＣＥ１Ｂ

レジスタ　ＣＣ，Ａ，Ｂ，Ｘ，Ｙ，Ｕ，Ｓ

解　説　このルーチンは，メモリ上にあるマシン語プログラムをセーブします．ファイルのフォーマットは下図のようになっています．

このルーチンの呼び出しは，アドレス情報をスタックに積んだ後ＪＭＰ命令によって行われる必要があります．しかし，ルーチンからはＲＴＳ命令でもどってくるため，あらかじめ，もどり番地をスタックに積んでおく必要があります．

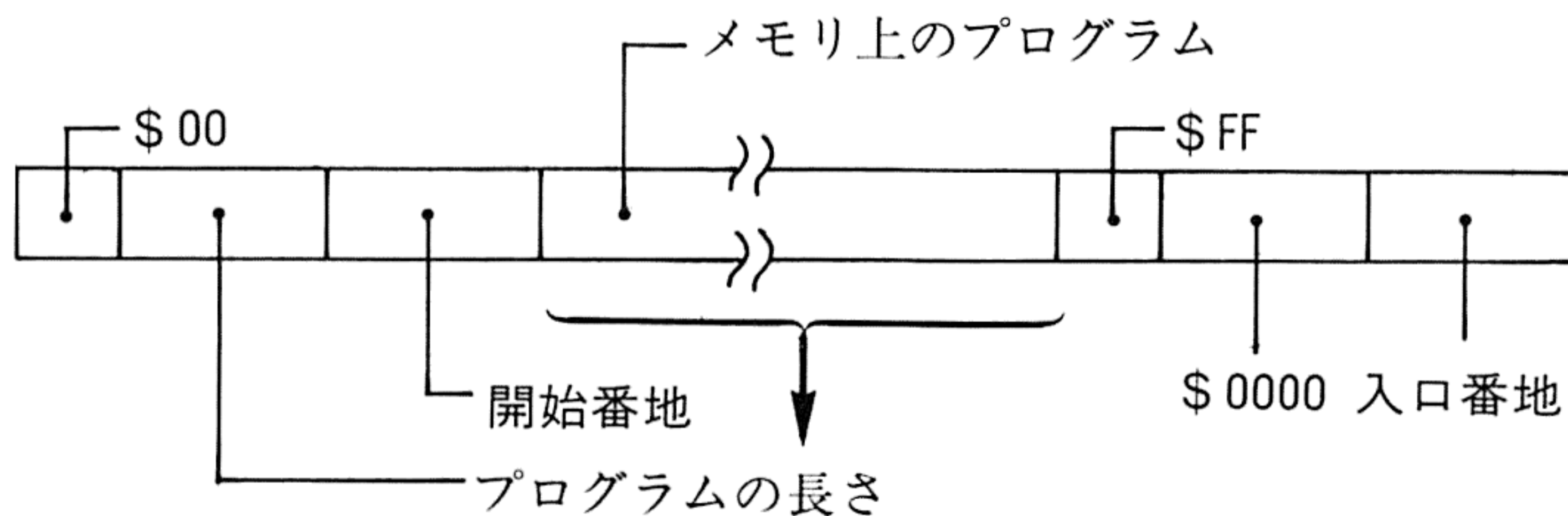
メモリ上のプログラム

＄00

＄FF

開始番地

＄0000　入口番地

プログラムの長さ

$CF77：プログラム・ロード

機　能	BASICまたはマシン語のプログラムをロードします．
パラメータ	$2F0←LOADMフラグ（0：LOADM，0以外LOAD） $2F1，$2F2←LOADMオフセット値（通常0） $2EE←即実行フラグ（3：即実行，0：即実行しない） $2EF←0 $2E6←物理機番 $2DD←ファイル名長 $2DE～$2E5←ファイル名
アドレス	$CF77（JSRで入ること）
レジスタ	CC，A，B，X，Y，U

解　説　LOADMフラグによって，BASICをロードするか，マシン語をロードするかを決定します．ロード中にエラーが生じた場合には，BASICのエラー処理を行い，コマンドレベルへ移ります．

BASICのプログラムをロードした場合には，BASICのコマンドレベルへ制御が移ります．マシン語プログラムをロードした場合には，RTSでもどってきます．

$CEDC, CEDF：ファイルオープン

機　能	ファイルのオープン処理を行います.
パラメータ	$BF←論理機番（ファイル番号）：1〜16（17は内部用）
	$2E6←物理機番（デバイス番号）：0〜$F, $80〜$83
	$2DD←ファイル名の長さ：0〜8
	$2DE〜$2E5←ファイル名（8文字に満たない分は空白）
	$2E7〜$2EC←ファイルオプション（最後は$00）
	$2DB←ファイルタイプ　}入力モードの時は，復帰情
	$2DC, $2D4←アスキーフラグ　}報となります.
アドレス	$CEDC：入力モード（OPEN"I"）のとき
	$CEDF：出力モード（OPEN"O"）のとき
レジスタ	Sを除いてすべて破壊される可能性があります.

解　説　物理機番で示されるデバイスにファイルをオープンし，ファイル番号と入出力バッファを割り当て，以下そのファイル番号での入出力を可能にします．ファイルオプションについては「文法書」2－18を御参照下さい.

エラーが生じた場合には，BASICのエラー処理を行います.

$CE4B：ファイルクローズ

機　能	ファイルのクローズ処理を行います.
パラメータ	$BF←論理機番：1〜16
アドレス	$CE4B（オープンされている全てのファイルをクローズしたいときは，$CE81．この場合，論理機番の指定は不要）
レジスタ	Sをのぞいてすべて破壊される可能性があります.

解　説　ファイル番号と出力バッファの割り当てを解除します．オープン同様エラーが生じた場合には，BASICのエラー処理を行います.

注）論理機番17は，BASIC内部で使用されているものなので，ユーザーは使用できません．また，入出力の際，論理機番に0を設定すると，出力時はスクリーン，入力時はキーボードが指定されます．論理機番0の入出力にはオープン処理は必要ありません.

物理機番対応表

入出力装置名	デバイス名	物理機番
キーボード	KYBD：	$00
スクリーン	SCRN：	$01
カセットテープ	CAS0：	$02
プリンタ	LPT0：	$03
RS232C	COM0：	$04
フロッピーディスク0	0：	$80
1	1：	$81
2	2：	$82
3	3：	$83

ファイルタイプ

	ファイルの形式
$00	BASICソース
$01	BASICデータ
$02	マシン語ファイル

アスキーフラグ

	ファイルの属性
$00	バイナリファイル
$FF	アスキーファイル

$D072：1文字入力

機　能　指定した論理機番のファイルから1文字入力します.

パラメータ　$BF←論理機番

アドレス　$D072

レジスタ　CC, A

復帰情報　Aレジスタ：入力した文字

$C0：エラーフラグ　≠0：エラーあり（Input past end他）
　　　　　　　　　=0：エラーなし

解　説　オープンされていない論理機番を指定した場合には，キーボードが割り当てられます．また，論理機番に0を指定した場合にもキーボードが割り当てられます.

キーボードからの入力の際に，画面へのエコーバックは行われません.

＄Ｄ０８Ｅ：１文字出力

機　能	**指定した論理機番のファイルから１文字入力します．**
パラメータ	**＄ＢＦ←論理機番**
	Ａレジスタ←出力する文字
アドレス	**＄Ｄ０８Ｅ**
レジスタ	**ＣＣ**

解　説　オープンされていない論理機番を指定した場合には，スクリーンが割り当てられます．

論理機番に０を指定した場合にもスクリーンが割り当てられます．この時＄５ＡＣの内容が０でないときには同時にプリンタにも出力されます．

指定した論理機番が，入力モードにオープンされている場合は，何も行いません．

＄Ｄ８０７：１行入力(1)

機　能	**指定した論理機番のファイルから１行入力を行います．**
パラメータ	**＄ＢＦ←論理機番**
アドレス	**＄Ｄ８０７**
レジスタ	**ＣＣ，Ａ，Ｂ，Ｘ，Ｙ，Ｕ**
復帰情報	**Ｘレジスタ＝＄４３Ｃ**
	＄４３Ｄ～＄５３Ｃ：入力文字列（＄０までが入力文字列）

解　説　このルーチンは１文字入力を利用して，指定ファイルから１ライン入力し，＄４３Ｄからのバッファに入力データを入れてもどります．

ラインインプットは，ＣＲ（＄０Ｄ）が入力されるか，ファイルの全データを入力し終るまで行われます．ただし，入力文字中のコントロールコード（＄０～＄１Ｆ）はバッファには転送されません（ＣＲも転送されません）．

リターン時には，Ｘレジスタにバッファアドレス－１が入っており，入力データの末尾には＄００を伴なっています．また，文字列の終りにスペースがある場合には，文字列後ろにスペースでない文字があるところまで，文字を削除しますので，文字列の最終文字は必ずスペース以外の文字となります．

＄D７F７：１行入力(2)

機　能　指定した論理機番のファイルから１行入力を行います．
パラメータ　＄BF←論理機番
アドレス　＄D７F７
レジスタ　CC，A，B，X，Y，U
復帰情報　Xレジスタ＝＄４３C
＄４３D～＄５３C：入力文字列（＄０までが入力文字列）

解　説　このルーチンは，BASICのスクリーンエディット用のルーチンで，入力した文字列の最初のキャラクタ（スペースはスキップする）が数字であった場合には次回このルーチンがコールされた時に，キー入力を行わずにまだ転送されていない変更フィールドを入力文字列として返します．しかし，これらの動作は，キーボードが割り当てられた場合のみで，それ以外のときは１行入力(1)とまったく同じです．

＄９BDB：１行出力(1)

機　能　指定した論理機番のファイルへ，１行出力を行います．
パラメータ　＄BF←論理機番
Xレジスタ←出力文字列の先頭アドレス－１
アドレス　＄９BDB
レジスタ　CC，A，B，X，Y，U

解　説　Xレジスタの内容＋１番地から，＄０又は＄２２（ダブルクォーテーション）を区切りとして文字列を出力します．

論理機番が０のときはスクリーンに出力され，さらに＄５ACの内容が０でない場合には，同時にプリンタにも出力されます．

＄９Ｃ１６：１行出力(2)

機　能　指定した論理機番のファイルへ，１行出力を行います．

パラメータ　＄ＢＦ←論理機番
Ｘレジスタ←出力文字列の先頭アドレス
Ｂレジスタ←文字列の長さ

アドレス　＄９Ｃ１６

レジスタ　ＣＣ，Ａ，Ｂ，Ｘ，Ｙ，Ｕ

解　説　Ｘレジスタの示す番地から，Ｂレジスタで示される文字数だけ出力します．

論理機番が０のときはスクリーンに出力され，さらに＄５ＡＣの内容が０でない場合には，同時にプリンタにも出力されます．

出力したい文字列の中にダブルクォーテーションをふくむ場合は，このルーチンを用いる必要があります．

＄９Ｂ５０：復改(1)

機　能　指定した論理機番のファイルへ，ＣＲとＬＦを出力します．

パラメータ　＄ＢＦ←論理機番

アドレス　＄９Ｂ５０

レジスタ　ＣＣ，Ａ，Ｂ，（指定する機器によっては，Ｙ，Ｕも破壊）

解　説　指定したファイルに対してＣＲとＬＦを出力して復改動作を行うわけですが，接続されている機器によって，ＣＲ，ＬＦのどちらか片方だけを出力する場合があります．この操作は，内部で行われます．

$ 9 B 4 7： 復改(2)

機　　能	指定した論理機番のファイルへ，ＣＲとＬＦを出力します．
パラメータ	＄ＢＦ←論理機番
アドレス	＄９Ｂ４７
レジスタ	ＣＣ，Ａ，Ｂ，（指定する機器によっては，Ｙ，Ｕも破壊）

解　　説　ファイルのポジション(BASICのＰＯＳで得られる値）が０でないときのみ復改動作（内容は復改(1)と同じ）を行います．

ただし，ディスク及びカセットテープには常に復改動作を行います．

◀ファイル入出力関係サンプル▶

ファイル入出力に関係するルーチンを使用したサンプルを２例示します．

1．アスキーセーブされたディスク上のファイルの内容を画面又はプリンタに出力します．

2．簡単なワードプロセッサです（このプログラムは後述のプリンタ関係のルーチンを使用しています）．

サンプル

```
PAGE   001  (         ,          )

01000                          *
01010                          * ASCII file output
01020                          *
01030                                  OPT     M,NOS,NOG,P=255
01040             9BDB         MESSAG  EQU     $9BDB    output from (X+1) till 0
01050             9C16         MESSA2  EQU     $9C16    output Breg bytes from (X)
01060             CEDF         OPENO   EQU     $CEDF    file open as output
01070             CEDC         OPENI   EQU     $CEDC    file open as input
01080             CE81         CLOSEA  EQU     $CE81    file close (ALL FILE)
01090             D072         IN      EQU     $D072    input 1 byte from file
01100             D08E         OUT     EQU     $D08E    output 1 byte to file
01110             9B50         CRLF    EQU     $9B50    output CR & LF
01120  5000                            ORG     $5000
01130             5000         START   EQU     *
01140                          *
01150                          * input file open
01160                          *
01170  5000 86    80                   LDA     #$80     device no. (DISK DRIVE 0)
01180  5002 B7    02E6                 STA     $2E6     set device no.
01190  5005 8D    79    5080           BSR     FILENA   get filename
01200  5007 86    01                   LDA     #1       set file no.
01210  5009 97    BF                   STA     $BF
01220  500B BD    CEDC                 JSR     OPENI    open as input
01230  500E B6    02DC                 LDA     $2DC     file ascii flag
01240  5011 81    FF                   CMPA    #$FF     ascii file ?
01250  5013 26    5E    5073           BNE     NOTASC   if not ascii file
01260                          *
01270                          * output file open
```

```
01280                          *
01290  5015 86    00                   LDA    #0        set file no.:SCRN & KYBD
01300  5017 97    BF                   STA    $BF
01310  5019 30    8D 00E1              LEAX   MES2-1,PCR output message
01320  501D BD    9BDB                 JSR    MESSAG
01330  5020 BD    D072         L00     JSR    IN        keyinput
01340  5023 C6    03                   LDB    #$03      device no. of LPT0:
01350  5025 81    70                   CMPA   #'p       output to printer ?
01360  5027 27    06    502F           BEQ    L01
01370  5029 C6    01                   LDB    #$01      device no. of SCRN:
01380  502B 81    73                   CMPA   #'s       output to screen ?
01390  502D 26    F1    5020           BNE    L00
01400  502F F7    02E6         L01     STB    $2E6      set device no.
01410  5032 86    02                   LDA    #2        set file no.
01420  5034 97    BF                   STA    $BF
01430  5036 4F                         CLRA             file name & option=null
01440  5037 B7    02DD                 STA    $2DD
01450  503A B7    02E7                 STA    $2E7
01460  503D BD    CEDF                 JSR    OPENO     open as output
01470                          *
01480                          * output file name
01490                          *
01500  5040 86    02                   LDA    #2        set file no.
01510  5042 97    BF                   STA    $BF
01520  5044 30    8D 00DA              LEAX   MES3-1,PCR output message
01530  5048 BD    9BDB                 JSR    MESSAG
01540  504B 30    8D 011C              LEAX   NAME,PCR load file name addr.
01550  504F C6    08                   LDB    #8        file name length
01560  5051 BD    9C16                 JSR    MESSA2    output
01570  5054 BD    9B50                 JSR    CRLF
01580                          *
01590                          * output ascii file
01600                          *
01610  5057 C6    01           L02     LDB    #1        set file no.
01620  5059 D7    BF                   STB    $BF
01630  505B BD    D072                 JSR    IN        input
01640  505E 0D    C0                   TST    $C0       test error flag
01650  5060 26    09    506B           BNE    L03       if input past end
01660  5062 C6    02                   LDB    #2        set file no.
01670  5064 D7    BF                   STB    $BF
01680  5066 BD    D08E                 JSR    OUT       output
01690  5069 20    EC    5057           BRA    L02       loop
01700  506B BD    CE81         L03     JSR    CLOSEA    close all file
01710  506E C6    00                   LDB    #0        file no. reset
01720  5070 D7    BF                   STB    $BF
01730  5072 39                         RTS
01740                          *
01750                          * if not ascii file then error
01760                          *
01770  5073 86    00           NOTASC  LDA    #0
01780  5075 97    BF                   STA    $BF
01790  5077 30    8D 00CF              LEAX   MES4-1,PCR
01800  507B BD    9BDB                 JSR    MESSAG
01810  507E 20    EB    506B           BRA    L03
01820                          *
01830                          * subroutine:set file name
01840                          *
01850  5080 86    00           FILENA  LDA    #0        ste file no.
01860  5082 97    BF                   STA    $BF
01870  5084 30    8D 004F              LEAX   MES1-1,PCR output message
01880  5088 BD    9BDB                 JSR    MESSAG
01890  508B CE    02DE                 LDU    #$2DE     file name address
01900  508E 30    8D 00D9              LEAX   NAME,PCR
01910  5092 5F                         CLRB             length counter
01920  5093 BD    D072         F01     JSR    IN        keyin !
01930  5096 81    08                   CMPA   #$08      back space ?
01940  5098 27    18    50B2           BEQ    BS
01950  509A 81    0D                   CMPA   #$0D      return key ?
01960  509C 27    2B    50C9           BEQ    CR
01970  509E 81    20                   CMPA   #$20      control code ?
01980  50A0 25    F1    5093           BCS    F01
01990  50A2 BD    D08E                 JSR    OUT       echo back
02000  50A5 A7    C0                   STA    ,U+       set
02010  50A7 A7    80                   STA    ,X+
```

```
02020  50A9 5C                     INCB
02030  50AA C1    08               CMPB   #8        length over 8 ?
02040  50AC 25    E5   5093        BCS    F01
02050  50AE F7    02DD             STB    $2DD      set length
02060  50B1 39                     RTS
02070  50B2 5D               BS    TSTB             if back space
02080  50B3 27    DE   5093        BEQ    F01
02090  50B5 BD    D08E             JSR    OUT
02100  50B8 86    20               LDA    #$20
02110  50BA BD    D08E             JSR    OUT
02120  50BD 86    08               LDA    #$08
02130  50BF BD    D08E             JSR    OUT
02140  50C2 33    5F               LEAU   -1,U
02150  50C4 30    1F               LEAX   -1,X
02160  50C6 5A                     DECB
02170  50C7 20    CA   5093        BRA    F01
02180  50C9 F7    02DD       CR    STB    $2DD      if CR
02190  50CC 86    20               LDA    #'        fill by space
02200  50CE A7    C0         C01   STA    ,U+
02210  50D0 A7    80               STA    ,X+
02220  50D2 5C                     INCB
02230  50D3 C1    08               CMPB   #8
02240  50D5 25    F7   50CE        BCS    C01
02250  50D7 39                     RTS
02260                        *
02270                        * messages
02280                        *
02290  50D8       0D0A       MES1  FDB    $0D0A
02300  50DA       2A               FCC    '** output ascii file **'
02310  50F1       0D0A             FDB    $0D0A
02320  50F3       20               FCC    ' FILE NAME:'
02330  50FE       00               FCB    0
02340  50FF       0D0A       MES2  FDB    $0D0A
02350  5101       20               FCC    ' output printer or screen ? (p/s)'
02360  5122       00               FCB    0
02370  5123       0D0A       MES3  FDB    $0D0A
02380  5125       2A               FCC    '*** ASCII FILE OUTPUT ***'
02390  513E       0D0A             FDB    $0D0A
02400  5140       46               FCC    'FILE NAME:'
02410  514A       00               FCB    0
02420  514B       0D0A       MES4  FDB    $0D0A
02430  514D       2A               FCC    '*** ERROR :not ASCII file !!!'
02440  516A       00               FCB    0
02450                        *
02460                        * file name save area
02470                        *
02480  516B       0008       NAME  RMB    8
02490                        *
02500             5000             END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5172
PROGRAM ENTRY ADDR=5000


exec &H5000

** output ascii file **
 FILE NAME:test!
 output printer or screen ? (p/s)
*** ASCII FILE OUTPUT ***
FILE NAME:test!

10 ' text program
20 '
30 A$=" text program here !!!"
40 PRINT A$

Ready
```

```
PAGE  001 (       ,        )

01000                          *
01010                          * micro word processer
01020                          *
01030                          *  !!! NOT POSITION INDEPENDENT !!!
01040                          *
01050            D08E          OUT    EQU    $D08E   output 1 character
01060            D807          LINEIN EQU    $D807   line input
01070            9B50          CRLF   EQU    $9B50   output CR & LF
01080            9BDB          MESSAG EQU    $9BDB   output from (X+1) till 0
01090            9C16          MESSA2 EQU    $9C16   output Breg bytes from (X)
01100            D9A7          PROPEN EQU    $D9A7   printer open
01110            D617          PROUT  EQU    $D617   printer output
01120            D64A          PRCRLF EQU    $D64A   printer CRLF
01130                                 OPT    M,NOG,NOS,P=255
01140  5000                           ORG    $5000
01150            5000          START  EQU    *
01160                          *
01170                          * cold start:$5000  hot start:$5002
01180                          *
01190  5000 20   02   5004            BRA    NEW
01200  5002 20   0B   500F            BRA    COMO
01210                          *
01220                          * new command :clear buffer
01230                          *
01240  5004 C6   41            NEW    LDB    #'A
01250  5006 8D   58   5060     NO1    BSR    ADDR
01260  5008 6F   84                   CLR    ,X
01270  500A 5C                        INCB
01280  500B C1   5A                   CMPB   #'Z
01290  500D 23   F7   5006            BLS    NO1
01300                          *
01310                          * command execution
01320                          *
01330  500F 8E   50EE          COMO   LDX    #MES1-1
01340  5012 4F                        CLRA
01350  5013 97   BF                   STA    $BF
01360  5015 BD   9BDB                 JSR    MESSAG
01370  5018 BD   D807          COM1   JSR    LINEIN  input line
01380  501B E6   01                   LDB    1,X     first chr.
01390  501D 27   F0   500F            BEQ    COMO
01400  501F CE   50E3                 LDU    #TABLE  command table
01410  5022 E1   C0            CO1    CMPB   ,U+
01420  5024 27   18   503E            BEQ    EXEC
01430  5026 33   42                   LEAU   2,U
01440  5028 1183 50EF                 CMPU   #.TABLE
01450  502C 26   F4   5022            BNE    CO1
01460  502E C1   41                   CMPB   #'A     input ?
01470  5030 25   04   5036            BCS    ERR
01480  5032 C1   5A                   CMPB   #'Z
01490  5034 23   0A   5040            BLS    INPUT
01500  5036 8E   5112          ERR    LDX    #MES2-1
01510  5039 BD   9BDB                 JSR    MESSAG
01520  503C 20   D1   500F            BRA    COMO
01530                          * command jump
01540  503E 6E   D4            EXEC   JMP    [,U]
01550                          *
01560                          * input line to buffer
01570                          *
01580  5040 33   02            INPUT  LEAU   2,X
01590  5042 8D   1C   5060            BSR    ADDR
01600  5044 A6   C0                   LDA    ,U+
01610  5046 81   3A                   CMPA   #':
01620  5048 26   EC   5036            BNE    ERR     if not *: then error
01630  504A 34   10                   PSHS   X
01640  504C 30   01                   LEAX   1,X
01650  504E 5F                        CLRB
01660  504F A6   C0            IO1    LDA    ,U+     transfer
01670  5051 27   07   505A            BEQ    IO2
01680  5053 A7   80                   STA    ,X+
01690  5055 5C                        INCB
01700  5056 C1   FF                   CMPB   #255
01710  5058 26   F5   504F            BNE    IO1
```

```
01720  505A 35    10          IO2    PULS   X
01730  505C E7    84                 STB    ,X
01740  505E 20    B8    5018         BRA    COM1
01750                         *
01760                         * subroutine:get buffer address of Breg line
01770                         *
01780  5060 34    06          ADDR   PSHS   D
01790  5062 C0    41                 SUBB   #'A
01800  5064 CB    30                 ADDB   #BUFFER/256
01810  5066 1F    98                 TFR    B,A
01820  5068 5F                       CLRB
01830  5069 1F    01                 TFR    D,X
01840  506B 35    86                 PULS   D,PC
01850                         *
01860                         * list command
01870                         *
01880  506D BD    9B50        LIST   JSR    CRLF
01890  5070 4F                       CLRA
01900  5071 97    BF                 STA    $BF
01910  5073 C6    41                 LDB    #'A
01920  5075 8D    E9    5060  LO1    BSR    ADDR
01930  5077 34    04                 PSHS   B
01940  5079 6D    84                 TST    ,X
01950  507B 27    14    5091         BEQ    LO2     if null
01960  507D 8D    21    50A0         BSR    FIELD   set field
01970  507F 1F    98                 TFR    B,A     output line no.
01980  5081 BD    D08E               JSR    OUT
01990  5084 86    3A                 LDA    #':
02000  5086 BD    D08E               JSR    OUT
02010  5089 E6    80                 LDB    ,X+     output line
02020  508B BD    9C16               JSR    MESSA2
02030  508E BD    9B50               JSR    CRLF
02040  5091 35    04          LO2    PULS   B
02050  5093 7D    0313               TST    $313    break key test
02060  5096 26    05    509D         BNE    LO3
02070  5098 5C                       INCB
02080  5099 C1    5A                 CMPB   #'Z
02090  509B 23    D8    5075         BLS    LO1
02100  509D 16    FF6F  500F  LO3    LBRA   COMO
02110                         *
02120                         * subroutine:set field (output $12,$07)
02130                         *
02140  50A0 34    16          FIELD  PSHS   D,X
02150  50A2 8E    5145               LDX    #SETF-1
02160  50A5 BD    9BDB               JSR    MESSAG
02170  50A8 35    96                 PULS   D,X,PC
02180                         *
02190                         * print command
02200                         *
02210  50AA CC    534E        PRINT  LDD    #'S*256+'N set file option
02220  50AD FD    02E7               STD    $2E7
02230  50B0 BD    D9A7               JSR    PROPEN  printer open
02240  50B3 C6    41                 LDB    #'A
02250  50B5 34    04          PO1    PSHS   B
02260  50B7 8D    A7    5060         BSR    ADDR
02270  50B9 E6    80                 LDB    ,X+
02280  50BB 27    12    50CF         BEQ    PO3
02290  50BD A6    80          PO2    LDA    ,X+     output a line
02300  50BF 34    10                 PSHS   X
02310  50C1 BE    05AA               LDX    $5AA
02320  50C4 BD    D617               JSR    PROUT
02330  50C7 35    10                 PULS   X
02340  50C9 5A                       DECB
02350  50CA 26    F1    50BD         BNE    PO2
02360  50CC BD    D64A               JSR    PRCRLF
02370  50CF 35    04          PO3    PULS   B
02380  50D1 5C                       INCB
02390  50D2 C1    5A                 CMPB   #'Z
02400  50D4 23    DF    50B5         BLS    PO1
02410  50D6 BD    D64A               JSR    PRCRLF
02420  50D9 16    FF33  500F         LBRA   COMO
```

```
02420  50D9 16    FF33  500F         LBRA   COMO
```

```
02430                         *
02440                         * exit command
```

```
02450                          *
02460  50DC 8E     512C        EXIT    LDX     #MES3-1
02470  50DF BD     9BDB                JSR     MESSAG
02480  50E2 39                         RTS
02490                          *
02500                          * command table
02510                          *
02520              50E3        TABLE   EQU     *
02530  50E3        2E                  FCC     '.'
02540  50E4        506D                FDB     LIST
02550  50E6        3F                  FCC     '?'
02560  50E7        50AA                FDB     PRINT
02570  50E9        23                  FCC     '#'
02580  50EA        5004                FDB     NEW
02590  50EC        21                  FCC     '!'
02600  50ED        50DC                FDB     EXIT
02610              50EF        .TABLE  EQU     *
02620                          *
02630                          * messages
02640                          *
02650  50EF        0D0A        MES1    FDB     $0D0A
02660  50F1        63                  FCC     'command [.,?,#,!] or [*:string]'
02670  5110        0D0A                FDB     $0D0A
02680  5112        00                  FCB     0
02690  5113        0D0A        MES2    FDB     $0D0A
02700  5115        2A                  FCC     '*** ERROR :bad command'
02710  512B        07                  FCB     $07,0
02720  512D        0D0A        MES3    FDB     $0D0A
02730  512F        21                  FCC     '!!! exit to system !!!
02740  5145        00                  FCB     0
02750  5146        12          SETF    FCB     $12,$07,0
02760                          *
02770                          * buffer $3000-$49FF (26*256=6656 bytes)
02780                          *
02790              3000        BUFFER EQU      $3000
02800                          *
02810              5000                END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5148
PROGRAM ENTRY ADDR=5000


  **** MICRO WORD PROCESSER ****
.
       << COMMAND LIST >>
            . :list
            ? :print
            # :new
            ! :exit
.
        << MEMORY USE >>
       $3000..$49FF :buffer
       $5000..$5148 :program
.
    << HOW TO INPUT LINES >>
ﾀﾄｴﾊﾞ 'A:This is a pen.' ﾄ ｷｰｲﾝ ｼﾀﾉﾁ retrun ｷｰ ｦ ｵｽ｡
.
        << HOW TO EDIT >>
'.'(list)ﾃﾞ ﾘｽﾄ ｦ ﾄｯﾀｱﾄ､ ｶｰｿﾙ ｦ ｲﾄﾞｳｼﾃ ﾃｲｾｲｽﾙ｡
  ( ﾀﾀﾞｼ ｲﾁﾄﾞﾆﾊ 1 ｷﾞｮｳｼｶ ﾃｲｾｲ ﾃﾞｷﾏｾﾝ｡ )
.
... ｺﾉ ﾌﾞﾝｼｮｳﾊ ｺﾉ ﾌﾟﾛｸﾞﾗﾑﾃﾞ output ｼﾏｼﾀ｡ ...
```

$AC37：Dレジスタ16進出力

機　能	Dレジスタの内容を16進数4ケタで出力します.
パラメータ	$BF←論理機番 Dレジスタ←出力したい値
アドレス	$AC37
レジスタ	CC, A

$AC3D：Aレジスタ16進出力

機　能	Aレジスタの内容を16進数2ケタで出力します.
パラメータ	$BF←論理機番 Dレジスタ←出力したい値
アドレス	$AC3D
レジスタ	CC, A

サンプル

```
PAGE   001 (         ,         )

01000                        *
01010                        * V-H check sum display
01020                        *
01030                               OPT     M,NOS,NOG,NOO,PAGE=255
01040              9BDB      MESSAG EQU     $9BDB   output from (X+1) till 0
01050              D08E      OUT    EQU     $D08E   one character output
01060              DB54      KEYIN  EQU     $DB54   keyinput with wait
01070              9B50      CRLF   EQU     $9B50   output CR & LF
01080              AC37      D16OUT EQU     $AC37   output Dreg in HEX
01090              AC3D      A16OUT EQU     $AC3D   output Areg in HEX
01100                        *
01110   5000                        ORG     $5000
01120              5000      START  EQU     *
01130                        *
01140                        * set display address
01150                        *
01160   5000 30    8D 0120          LEAX    MES1-1,PCR
01170   5004 BD    9BDB             JSR     MESSAG
01180   5007 17    00E0 50EA        LBSR    GETADR
01190   500A C4    F0               ANDB    #$F0    mask LSN
01200   500C 34    06               PSHS    D
01210   500E 30    8D 0125          LEAX    MES2-1,PCR
01220   5012 BD    9BDB             JSR     MESSAG
01230   5015 17    00D2 50EA        LBSR    GETADR
01240   5018 ED    8D 01A7          STD     ENDADR,PCR
01250   501C 30    8D 012A          LEAX    MES3-1,PCR
01260   5020 BD    9BDB             JSR     MESSAG
01270   5023 BD    DB54             JSR     KEYIN
01280   5026 81    79               CMPA    #'y
01290   5028 26    03    502D       BNE     BLOCK
```

```
01300  502A 7C    05AC             INC     $5AC      set printer flag
01310                       *
01320                       * 1 block (256bytes) output
01330                       *
01340  502D 35    10        BLOCK  PULS    X
01350  502F AC    8D 0190          CMPX    ENDADR,PCR
01360  5033 1022  00AF 50E6        LBHI    END
01370  5037 33    8D 0178          LEAU    VSUM,PCR
01380  503B C6    0F               LDB     #$0F
01390  503D 6F    C5        B01    CLR     B,U
01400  503F 5A                     DECB
01410  5040 2A    FB   503D        BPL     B01
01420  5042 34    10               PSHS    X
01430  5044 30    8D 0123          LEAX    MES4-1,PCR
01440  5048 BD    9BDB             JSR     MESSAG
01450  504B 35    10               PULS    X
01460                       *
01470                       * 1 line (16bytes) output
01480                       *
01490  504D BD    9B50      LINE   JSR     CRLF
01500  5050 1F    10               TFR     X,D
01510  5052 BD    AC37             JSR     D16OUT
01520  5055 86    20               LDA     #'        space
01530  5057 BD    D08E             JSR     OUT
01540  505A 5F                     CLRB
01550  505B 33    8D 0154          LEAU    VSUM,PCR
01560  505F 6F    8D 014F          CLR     HSUM,PCR
01570                       *
01580                       * 1 byte output
01590                       *
01600  5063 A6    80        BYTE   LDA     ,X+
01610  5065 34    02               PSHS    A
01620  5067 AB    8D 0147          ADDA    HSUM,PCR
01630  506B A7    8D 0143          STA     HSUM,PCR
01640  506F A6    E4               LDA     ,S
01650  5071 AB    C5               ADDA    B,U
01660  5073 A7    C5               STA     B,U
01670  5075 35    02               PULS    A
01680  5077 BD    AC3D             JSR     A16OUT
01690  507A 86    20               LDA     #'        space
01700  507C BD    D08E             JSR     OUT
01710  507F 5C                     INCB
01720  5080 C1    10               CMPB    #$10
01730  5082 26    DF   5063        BNE     BYTE
01740                       *
01750  5084 86    3A        NEXTL  LDA     #':
01760  5086 BD    D08E             JSR     OUT
01770  5089 A6    8D 0125          LDA     HSUM,PCR
01780  508D BD    AC3D             JSR     A16OUT
01790  5090 1F    10               TFR     X,D
01800  5092 5D                     TSTB
01810  5093 27    08   509D        BEQ     NEXTB
01820  5095 AC    8D 012A          CMPX    ENDADR,PCR
01830  5099 22    02   509D        BHI     NEXTB
01840  509B 20    B0   504D        BRA     LINE
01850                       *
01860  509D BD    9B50      NEXTB  JSR     CRLF
01870  50A0 86    2D               LDA     #'-
01880  50A2 C6    39               LDB     #57
01890  50A4 BD    D08E      NB1    JSR     OUT
01900  50A7 5A                     DECB
01910  50A8 26    FA   50A4        BNE     NB1
01920  50AA 6F    8D 0104          CLR     HSUM,PCR
01930  50AE 34    10               PSHS    X
01940  50B0 30    8D 00F5          LEAX    MES5-1,PCR
01950  50B4 BD    9BDB             JSR     MESSAG
01960  50B7 33    8D 00F8          LEAU    VSUM,PCR
01970  50BB 5F                     CLRB
01980  50BC A6    C5        NB2    LDA     B,U
01990  50BE 34    02               PSHS    A
02000  50C0 AB    8D 00EE          ADDA    HSUM,PCR
02010  50C4 A7    8D 00EA          STA     HSUM,PCR
02020  50C8 35    02               PULS    A
02030  50CA BD    AC3D             JSR     A16OUT
```

```
02040   50CD 86   20                 LDA    #        space
02050   50CF BD   D0BE               JSR    OUT
02060   50D2 5C                      INCB
02070   50D3 C1   10                 CMPB   #$10
02080   50D5 26   E5    50BC         BNE    NB2
02090   50D7 86   3A                 LDA    #':
02100   50D9 BD   D0BE               JSR    OUT
02110   50DC A6   8D 00D2            LDA    HSUM,PCR
02120   50E0 BD   AC3D               JSR    A16OUT
02130   50E3 16   FF47 502D          LBRA   BLOCK
02140                         *
02150                         * end of display
02160                         *
02170   50E6 7F   05AC        END    CLR    $5AC
02180   50E9 39                      RTS
02190                         *
02200                         * subroutine:input HEX ADDR in D reg
02210                         *
02220   50EA CC   0000        GETADR LDD    #0
02230   50ED 34   06                 PSHS   D
02240   50EF 34   06          GA01   PSHS   D        save keyin
02250   50F1 EC   62                 LDD    2,S      Dreg=Dreg*16+keyin
02260   50F3 58                      ASLB
02270   50F4 49                      ROLA
02280   50F5 58                      ASLB
02290   50F6 49                      ROLA
02300   50F7 58                      ASLB
02310   50F8 49                      ROLA
02320   50F9 58                      ASLB
02330   50FA 49                      ROLA
02340   50FB E3   E1                 ADDD   ,S++
02350   50FD ED   E4                 STD    ,S
02360   50FF BD   DB54               JSR    KEYIN    keyinput
02370   5102 81   30                 CMPA   #'0      test '0'..'9' or 'A'..'F'
02380   5104 25   1D    5123         BCS    GA04
02390   5106 81   39                 CMPA   #'9
02400   5108 22   07    5111         BHI    GA02
02410   510A BD   D0BE               JSR    OUT
02420   510D 80   30                 SUBA   #'0
02430   510F 20   0D    511E         BRA    GA03
02440   5111 81   41          GA02   CMPA   #'A
02450   5113 25   0E    5123         BCS    GA04
02460   5115 81   46                 CMPA   #'F
02470   5117 22   0A    5123         BHI    GA04
02480   5119 BD   D0BE               JSR    OUT
02490   511C 80   37                 SUBA   #'A-10
02500   511E 1F   89          GA03   TFR    A,B
02510   5120 4F                      CLRA
02520   5121 20   CC    50EF         BRA    GA01
02530   5123 35   86          GA04   PULS   D,PC     address in Dreg
02540                         *
02550                         * messages
02560                         *
02570   5125      0D0A        MES1   FDB    $0D0A
02580   5127      53                 FCC    'START ADDRESS =$'
02590   5137      00                 FCB    0
02600   5138      0D0A        MES2   FDB    $0D0A
02610   513A      45                 FCC    'END   ADDRESS =$'
02620   514A      00                 FCB    0
02630   514B      0D0A        MES3   FDB    $0D0A
02640   514D      20                 FCC    ' output to printer ? (y/else)
02650   516B      00                 FCB    0
02660   516C      0D0A        MES4   FDB    $0D0A,$0D0A
02670   5170      41                 FCC    'ADD  +0 +1 +2 +3 +4 +5 +6 +7'
02680   518C      20                 FCC    ' +8 +9 +A +B +C +D +E +F :SUM'
02690   51A9      00                 FCB    0
02700   51AA      0D0A        MES5   FDB    $0D0A
02710   51AC      53                 FCC    'SUM
02720   51B1      00                 FCB    0
02730                         *
02740                         * working area
02750                         *
02760   51B2      00          HSUM   FCB    0
02770   51B3      0010        VSUM   RMB    16
```

```
02780  51C3         0000       ENDADR FDB     0
02790                          *
02800               5000              END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=51C4
PROGRAM ENTRY ADDR=5000


exec &H5000

START ADDRESS =$5000
END   ADDRESS =$51C4
 output to printer ? (y/else)
ADD  +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F :SUM
5000 30 8D 01 20 BD 9B DB 17 00 E0 C4 F0 34 06 30 8D :B3
5010 01 25 BD 9B DB 17 00 D2 ED 8D 01 A7 30 8D 01 2A :4C
5020 BD 9B DB BD DB 54 81 79 26 03 7C 05 AC 35 10 AC :60
5030 BD 01 90 10 22 00 AF 33 8D 01 78 C6 0F 6F C5 5A :9B
5040 2A FB 34 10 30 8D 01 23 BD 9B DB 35 10 BD 9B 50 :6A
5050 1F 10 BD AC 37 86 20 BD D0 8E 5F 33 8D 01 54 6F :73
5060 8D 01 4F A6 80 34 02 AB 8D 01 47 A7 8D 01 43 A6 :D7
5070 E4 AB C5 A7 C5 35 02 BD AC 3D 86 20 BD D0 8E 5C :BA
5080 C1 10 26 DF 86 3A BD D0 8E A6 8D 01 25 BD AC 3D :B0
5090 1F 10 5D 27 08 AC 8D 01 2A 22 02 20 B0 BD 9B 50 :BB
50A0 86 2D C6 39 BD D0 8E 5A 26 FA 6F 8D 01 04 34 10 :8C
50B0 30 8D 00 F5 BD 9B DB 33 8D 00 F8 5F A6 C5 34 02 :9D
50C0 AB 8D 00 EE A7 8D 00 EA 35 02 BD AC 3D 86 20 BD :84
50D0 D0 8E 5C C1 10 26 E5 86 3A BD D0 8E A6 8D 00 D2 :76
50E0 BD AC 3D 16 FF 47 7F 05 AC 39 CC 00 00 34 06 34 :A5
50F0 06 EC 62 58 49 58 49 58 49 58 49 E3 E1 ED E4 BD :2A
---------------------------------------------------------
SUM  09 92 72 E2 48 25 90 08 35 EA 58 BB 46 3D 7F 9D :C5

ADD  +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F :SUM
5100 DB 54 81 30 25 1D 81 39 22 07 BD D0 8E 80 30 20 :F0
5110 0D 81 41 25 0E 81 46 22 0A BD D0 8E 80 37 1F 89 :6F
5120 4F 20 CC 35 86 0D 0A 53 54 41 52 54 20 41 44 44 :84
5130 52 45 53 53 20 3D 24 00 0D 0A 45 4E 44 20 20 20 :0C
5140 41 44 44 52 45 53 53 20 3D 24 00 0D 0A 20 6F 75 :A2
5150 74 70 75 74 20 74 6F 20 70 72 69 6E 74 65 72 20 :14
5160 3F 20 28 79 2F 65 6C 73 65 29 20 00 0D 0A 0D 0A :4F
5170 41 44 44 20 20 2B 30 20 2B 31 20 2B 32 20 2B 33 :DB
5180 20 2B 34 20 2B 35 20 2B 36 20 2B 37 20 2B 38 20 :A5
5190 2B 39 20 2B 41 20 2B 42 20 2B 43 20 2B 44 20 2B :E5
51A0 45 20 2B 46 20 3A 53 55 4D 00 0D 0A 53 55 4D 20 :51
51B0 20 00 20 6E D6 A5 3B EF 73 2C 32 E0 76 7A E7 43 :1E
51C0 05 58 8D 51 C4 00 00 00 00 00 00 00 00 00 00 00 :FF
---------------------------------------------------------
SUM  73 2E 32 8C B3 73 2C 32 E0 76 7A E7 43 05 58 8D :C7
```

＄８６５６：画面表示関係初期化

機　能	主にサブシステムに関係するパラメータを初期化します．
パラメータ	＄１Ｅ６←背景色（０～７） ＄Ｃ３←表示桁数（４０，８０） ＄３０Ｄ←表示行数（２０，２５） ＄３０Ｅ←スクロール開始行（０～２４） ＄３０Ｆ←スクロール終了行（０～２４） ＄３１０←ファンクションキー表示フラグ（０：非表示） ＄５ＡＤ←単色表示フラグ（０：非指定，０以外：単色指定）
アドレス	＄８６５６
レジスタ	ＣＣ，Ａ，Ｂ，Ｘ，Ｙ，Ｕ

解　説　具体的には以下のことを行います．

- ファンクションキーを，ＲＯＭモード用に初期化します．
- ＴＡＢを８文字ごとに設定します．
- カーソル表示，オーダ動作，ＴＡＢ動作，ページウエイト，プットウエイトを指定します．オートＬＦは指定しません．
- タイマを０時割り込みのみ可能とします．
- パラメータ の項の設定にしたがって画面を初期化します．
- 論理機番を０に設定します．
- キーの先行入力を禁止します．

＄Ｃ８ＢＣ：画面表示設定

機　能	画面表示に関する設定を行います．
パラメータ	上項（＄８６５６：画面表示関係初期化）と同様です．
アドレス	＄Ｃ８ＢＣ
レジスタ	ＣＣ，Ａ，Ｂ，Ｘ，Ｕ

解　説　このルーチンは，サブシステムコマンドのＩＮＩＴコマンドを，ＢＩＯＳ経由で利用することにより，画面表示に関する設定を行います．

＄ＤＡＦ９：画面機能設定

機　能	画面機能を設定します．
パラメータ	Ｂレジスタ←サブシステムコマンドＣＮＳＣＴＬの制御フラグ

bit 5　オートＬＦ
4　プットウェイト
3　ページウェイト
2　ＴＡＢ動作
1　オーダー動作
0　カーソル表示

（bit 5～0）する＝1，しない＝0

アドレス	＄ＤＡＦ９
レジスタ	ＣＣ，Ａ

解　説	サブシステムコマンドＣＮＳＣＴＬを利用し画面機能を設定します．

＄ＤＡＦ６：カーソル表示

機　能	カーソルを表示します．
アドレス	＄ＤＡＦ６
レジスタ	ＣＣ，Ａ，Ｂ

解　説　上項（＄ＤＡＦ９：画面機能設定）を利用して，カーソルの表示をサブシステムに指示します．

具体的には，＄５Ａ８の内容をＢレジスタにセットして上項のルーチンを実行します．このため，ＬＯＣＡＴＥ文でカーソルが表示されないようにしてある場合（＄５Ａ８の bit 0 が 0）には，カーソルは表示しません．

＄ＤＡＥＦ：カーソル消去

機　能	カーソルを消去します．
アドレス	＄ＤＡＥＦ
レジスタ	ＣＣ，Ａ，Ｂ

解　説　カーソル表示同様，「画面機能設定」のルーチンを使用します．

具体的には，＄５Ａ８の内容をＢレジスタにセットし，bit 0 を 0 にした後（ＡＮＤＢ＃＄ＦＥ），＄ＤＡＦ９からのルーチンを実行します．

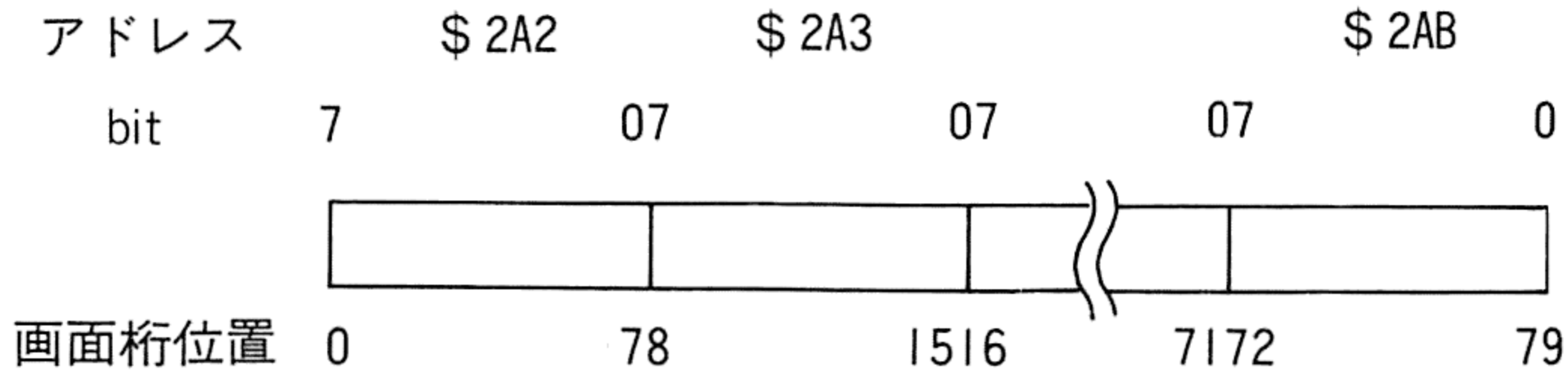
サンプル

```
10 '
20 ' get number without INPUT
30 '             (not display '? REDO ..')
40 PRINT
50 PRINT "input number = ";
60 GOSUB 10000
70 PRINT:PRINT "input number =";NUMBER
80 IF NUMBER=0 THEN END ELSE 40
10000 '
10010 ' subroutine:get number
10020 '                 input : non
10030 ' destory variables : NUMBER,COLUMN,DOT,INK$,INK
10040 '               output : RESULT$ = input sting
10050 '                        NUMBER  = input number
10060 '
10070 EXEC &HDAF6:' corsor on
10080 RESULT$="":COLUMN=0:DOT=-1
10090 INK$=INKEY$:IF INK$="" THEN 10090
10100 INK=ASC(INK$)
10110 IF INK=&H08 THEN GOSUB 10230:GOTO 10090:' BS
10120 IF INK=&H7F THEN GOSUB 10230:GOTO 10090:' DEL
10130 IF INK=&H0D THEN EXEC &HDAF6:NUMBER=VAL(RESULT$):RETURN:'corsor off
10140 IF INK$="." THEN IF DOT=-1 THEN DOT=COLUMN+1:GOTO 10170 ELSE 10090
10150 IF INK$="-" OR INK$="+" THEN IF COLUMN=0 THEN 10170 ELSE 10090
10160 IF INK<&H30 OR INK>&H39 THEN 10090: '1'..'9'
10170 BEEP 1
10180 COLUMN=COLUMN+1
10190 RESULT$=RESULT$+INK$
10200 BEEP 0
10210 PRINT INK$;
10220 GOTO 10090
10230 ' BS or DEL
10240 IF COLUMN=0 THEN RETURN
10250 IF COLUMN=DOT THEN DOT=-1
10260 COLUMN=COLUMN-1:RESULT$=LEFT$(RESULT$,COLUMN)
10270 PRINT CHR$(29);" ";CHR$(29);
10280 RETURN


run

input number = 1.2345
input number = 1.2345

input number = -12345.45
input number =-12345.5

input number = +.1
input number = .1

input number =
input number = 0

Ready
```

＄８６Ａ６：ＴＡＢセット

機　能　ＴＡＢ動作におけるカーソル停止位置を設定します．

パラメータ　＄２Ａ２～＄２ＡＢ：ＴＡＢセットマップ

アドレス	＄2A2	＄2A3	…	＄2AB
bit	7　　　0	7　　　0	7	0
画面桁位置	0　　　7	8　　　15	16 … 71	72　　　79

アドレス　＄８６Ａ６

レジスタ　ＣＣ，Ａ，Ｂ，Ｘ

解　説　サブシステムコマンドＴＡＢＳＥＴを利用して，ＴＡＢ位置を設定します．

サンプル

```
1 '
2 ' tab set
3 '
4 ' command :               <- , -> :cursor move
5 '           shift <-,shift -> :cursor move
6 '                         CR : set position
7 '                      space : reset position
8 '                        ESC : end of set operation
100 DEFINT A-Z:DIM BIT(7):TAB=&H2A2
110 ' display tab posistion
120 PRINT:FOR I=0 TO 7:PRINT "+123456789";:NEXT
130 FOR I=TAB TO TAB+9
140  BYTE=PEEK(I)
150  FOR B=0 TO 7:BIT(B)=SGN((2^B) AND BYTE):NEXT
160  FOR B=7 TO 0 STEP -1
170   IF BIT(B)=0 THEN PRINT"-"; ELSE PRINT"*";
180  NEXT
190 NEXT:PRINT:PRINT:PRINTCHR$(30);CHR$(30);
200 V=CSRLIN:H=0
210 LOCATE H,V:PRINT"~";
220 A$=INPUT$(1):LOCATE H,V:PRINT" ";
230 IF A$=CHR$(28) THEN H=H+1:IF H>79 THEN H=0:GOTO 210 ELSE 210
240 IF A$=CHR$(6) THEN H=H+8:IF H>79 THEN H=79:GOTO 210 ELSE 210
250 IF A$=CHR$(29) THEN H=H-1:IF H<0 THEN H=79:GOTO 210 ELSE 210
260 IF A$=CHR$(2) THEN H=H-8:IF H<0 THEN H=0:GOTO 210 ELSE 210
270 IF A$=CHR$(13) THEN LOCATE H,V-1:PRINT"*";:GOTO 210
280 IF A$=CHR$(32) THEN LOCATE H,V-1:PRINT"-";:GOTO 210
290 IF A$=CHR$(27) THEN 310
300 GOTO 210
310   end of set
320 FOR I=0 TO 9
330  BYTE=0
340  FOR B=7 TO 0 STEP -1:H=7-B+I*8
350   FLAG=SCREEN(H,V-1)
360   IF FLAG=ASC("*") THEN BYTE=BYTE+2^B
370  NEXT
380  POKE &H2A2+I,BYTE
390 NEXT
```

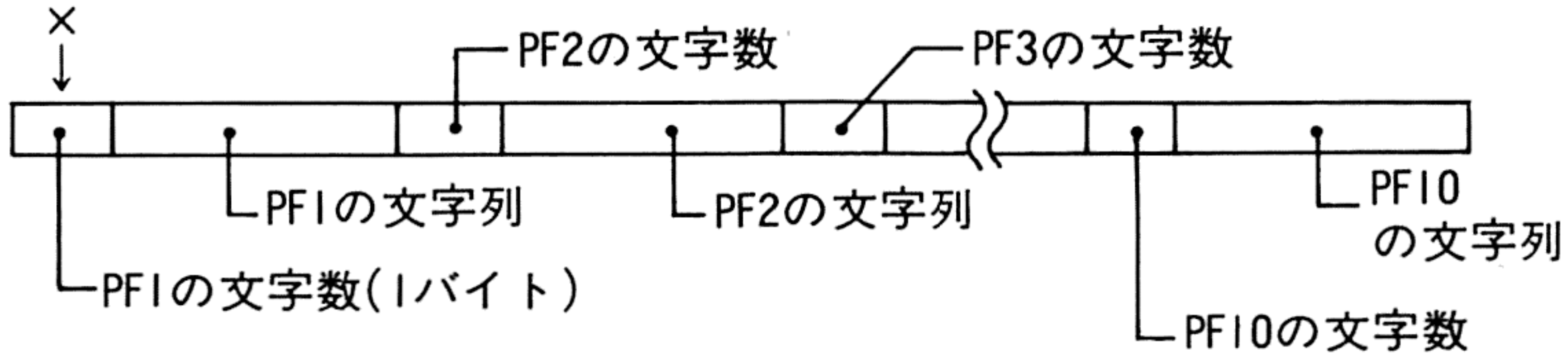
```
400 EXEC &H86A6
410 END
420 FOR Z1=0 TO 7:BIT(Z1)=SGN((2^Z1)AND BYTE):NEXT:RETURN

RUN

+123456789+123456789+123456789+123456789+123456789+123456789+123456789+123456789
----------*---------*---------*---------*---------------------------------------

Ready
for i=1 to 10:? chr$(9);"!";:next
          !         !         !         !
!         !         !         !         !
!
Ready
```

$８６ＣＦ：ＰＦキー設定

機　能　ＰＦキーを１０個分まとめて設定します．

パラメータ　Ｘレジスタ←データ先頭アドレス

データのフォーマット

X
PF2の文字数
PF3の文字数
PF1の文字列
PF2の文字列
PF10
の文字列
PF1の文字数(1バイト)
PF10の文字数

アドレス　$８６ＣＦ

レジスタ　ＣＣ，Ａ，Ｂ，Ｘ，Ｙ，Ｕ

解　説　ＰＦキーを１０個分まとめて設定します．アボート時（リセット時でＲＯＭモードの時も含む）には，ＰＦキーを下記のデータで，このルーチンを使用して設定します．

```
 X ﾚｼﾞｽﾀ =$876B
 addr:byte characters
 ress

$876B: 05 41 55 54 4F 20
          A  U  T  O
$8771: 05 4C 49 53 54 0D
          L  I  S  T  .
$8777: 04 52 55 4E 0D
          R  U  N  .
$877C: 05 43 4F 4E 54 0D
          C  O  N  T  .
$8782: 06 4C 4C 49 53 54 0D
          L  L  I  S  T  .
$8789: 05 4C 4F 41 44 0D
          L  O  A  D  .
$878F: 05 53 41 56 45 22
          S  A  V  E  "
$8795: 0D.3F 44 41 54 45 24 2C 54 49 4D 45 24 0D
          ?  D  A  T  E  $  ,  T  I  M  E  $  .
$87A3: 0A 53 43 52 45 45 4E 37 2C 37 0D
          S  C  R  E  E  N  7  ,  7  .
$87AE: 06 48 41 52 44 43 0D
          H  A  R  D  C  .
```

＄Ｄ９Ｄ９：画面１文字出力

機　　能　画面に１文字出力します.

パラメータ　Ａレジスタ←出力文字コード

アドレス　＄Ｄ９Ｄ９

レジスタ　ＣＣ

解　　説　ＢＩＯＳ経由でサブシステムコマンドＰＵＴを使用することにより，画面に１文字出力します.

＄ＤＢ５４：キーボード１文字入力(1)

機　　能　キーボードから１文字入力します（キー入力があるまで待ちます).

アドレス　＄ＤＢ５４

レジスタ　ＣＣ，Ａ，Ｂ，Ｘ

復帰情報　Ａレジスタ：入力された文字のコード

解　　説　ＢＩＯＳを使用して，キーボードから１文字入力します.

＄ＤＢ６Ｄ：キーボード１文字入力(2)

機　　能　キーボードから１文字入力します．ただし，キーボードバッファが空の場合は，＄００を返します.

アドレス　＄ＤＢ６Ｄ

レジスタ　ＣＣ，Ａ，Ｂ，Ｘ

復帰情報　ＣＣレジスタＺフラグ：１：キーボードバッファが空
　　　　　　　　　　　　　　　　０：入力された文字がある
　　　　　Ａレジスタ：入力された文字コード（ただしＺ＝１のときは，＄００がセットされます.）

解　　説　ＢＩＯＳを使用して，キーボードから１文字入力します.

注）ここに示した２つのキーボード１文字入力は，その操作をキーボードバッファに対して行うので注意が必要です.

＄DA6D：カーソル位置設定

機　能	カーソル位置を設定します．
パラメータ	Aレジスタ←桁（0～39 or 79） Bレジスタ←行（0～19 or 24）
アドレス	＄DA6D
レジスタ	CC，A

解　説　カーソル位置を，サブシステムコマンドPUTのオーダ＄12によって，設定します．

＄DA8B：画面消去

機　能	指定した画面を消去します．
パラメータ	Bレジスタ←消去画面コード　0：全画面 1：スクロールモード画面 2：ページモード1画面 3：ページモード2画面 ＄1E5←文字色 ＄1E6←背景色（全画面消去を指定した場合のみ有効）
アドレス	＄DA8B（全画面を指定する場合には，パラメータなしで＄DA8Aを呼んでもよい）
レジスタ	A，（＄DA8Aを呼んだ場合はBも破壊）

解　説　サブシステムコマンドERASE2を使用して，画面を消去します．

◀画面・キーボード関係サブルーチン・サンプル▶

画面1文字出力，キーボード1文字入力(2)，カーソル位置設定を利用した簡単なゲームです（このプログラムは，ポジションインディペンデントではありません）．

サンプル

```
10 ' sample game program
20 '           1 player TV-TENNIS
30 '
40 '                     presented by H.NAKAMURA
50 '
100 CLEAR,&H4FFF
110 WIDTH 40,25
120 LOCATE0,24:INPUT "GAME LEVEL (0:FAST..10:SLOW) ";LEVEL
130 IF LEVEL<0 OR LEVEL>10 THEN 120
140 LEVEL=LEVEL*1000+1
150 POKE &H5100,1 :POKE &H5101,256-1:'dx,dy
160 POKE &H5102,10:POKE &H5103,10:' old x,y
170 POKE &H5104,10:POKE &H5105,10:' new x,y
180 POKE &H5106,20:POKE &H5107,0 :' pd ,pdd
190 POKE &H5108, 0:POKE &H5109,0 :' score
200 POKE &H510A,LEVEL ¥ 256:POKE &H510B, LEVEL MOD 256:' wait constant
210 CLS
220 LINE(0,0)-(38,0),"■":LINE(0,0)-(0,23),"■":LINE(38,0)-(38,23),"■"
230 EXEC &H5000
240 LOCATE 15,10:PRINT"SCORE=";PEEK(&H5108)*256+PEEK(&H5109)
250 FOR I=0 TO 3000:NEXT
260 PRINT CHR$(&H1B)+CHR$(&H39)
270 GOTO 120
```

```
PAGE  001 (        ,        )

01000                              *
01010                              * sample game program
01020                              *                  presented by H.NAKAMURA
01030                                     OPT     M,NOS,NOG,P=255
01040                D9D9         SOUT    EQU     $D9D9    display 1 byte
01050                DB6D         INKEY   EQU     $DB6D    keyinput without wait
01060                DA6D         LOCATE  EQU     $DA6D    locate x,y
01070    5000                             ORG     $5000
01080                5000         START   EQU     *
01090                              *
01100                              * main program
01110                              *
01120    5000 8D     1B    501A   MAIN    BSR     XMOVE    x-dir move
01130    5002 8D     2B    502F           BSR     YMOVE    y-dir move
01140    5004 34     01                   PSHS    CC
01150    5006 8D     6F    5077           BSR     DSP      display ball
01151    5008 35     01                   PULS    CC
01155    500A 25     0A    5016           BCS     MO1      miss ?
01160    500C 17     0081  5090           LBSR    PDMOVE
01170    500F 8D     7F    5090           BSR     PDMOVE   paddle move
01180    5011 17     00CA  50DE           LBSR    WAIT     wait
01190    5014 20     EA    5000           BRA     MAIN     repeat
01200    5016 39                  MO1     RTS
01210                              *
01220    5017 70     5100         XMO1    NEG     DX
01230    501A B6     5104         XMOVE   LDA     XX
01240    501D BB     5100                 ADDA    DX
01250    5020 27     F5    5017           BEQ     XMO1
01260    5022 2B     F3    5017           BMI     XMO1
01270    5024 81     26                   CMPA    #38
01280    5026 24     EF    5017           BCC     XMO1
01290    5028 B7     5104                 STA     XX
01300    502B 39                          RTS
01310                              *
01320    502C 70     5101         YMO1    NEG     DY
01330    502F B6     5105         YMOVE   LDA     YY
01340    5032 BB     5101                 ADDA    DY
01350    5035 27     F5    502C           BEQ     YMO1
01360    5037 B7     5105                 STA     YY
01370    503A 81     18                   CMPA    #24
01380    503C 27     03    5041           BEQ     YMO2
```

```
01390  503E 1C   FE                  ANDCC   #$FE
01400  5040 39                       RTS
01410  5041 70   5101         YM02   NEG     DY
01420  5044 B6   5106                LDA     PD
01430  5047 B0   5104                SUBA    XX
01440  504A 34   02                  PSHS    A
01450  504C B6   5100                LDA     DX
01460  504F A0   E4                  SUBA    ,S
01470  5051 81   FE                  CMPA    #-2
01480  5053 2D   04     5059         BLT     YM03
01490  5055 81   02                  CMPA    #2
01500  5057 2F   02     505B         BLE     YM04
01510  5059 AB   E4           YM03   ADDA    ,S
01520  505B B7   5100         YM04   STA     DX
01530  505E 35   02                  PULS    A
01540  5060 81   FE                  CMPA    #-2
01550  5062 2D   10     5074         BLT     YM05
01560  5064 81   02                  CMPA    #2
01570  5066 2E   0C     5074         BGT     YM05
01580  5068 FC   5108                LDD     SC
01590  506B C3   000A                ADDD    #10
01600  506E FD   5108                STD     SC
01610  5071 1C   FE                  ANDCC   #$FE
01620  5073 39                       RTS
01630  5074 1A   01           YM05   ORCC    #$01
01640  5076 39                       RTS
01650                         *
01660  5077 FC   5102         DSP    LDD     OLDX
01670  507A BD   DA6D                JSR     LOCATE
01680  507D 86   20                  LDA     #'        space
01690  507F BD   D9D9                JSR     SOUT
01700  5082 FC   5104                LDD     XX
01710  5085 FD   5102                STD     OLDX
01720  5088 BD   DA6D                JSR     LOCATE
01730  508B 86   EC                  LDA     #'●
01740  508D 7E   D9D9                JMP     SOUT
01750                         *
01760  5090 BD   DB6D         PDMOVE JSR     INKEY
01770  5093 27   1A     50AF         BEQ     PDM03
01780  5095 81   34                  CMPA    #'4
01790  5097 26   07     50A0         BNE     PDM01
01800  5099 86   FF                  LDA     #-1
01810  509B B7   5107                STA     PDD
01820  509E 20   0F     50AF         BRA     PDM03
01830  50A0 81   36           PDM01  CMPA    #'6
01840  50A2 26   07     50AB         BNE     PDM02
01850  50A4 86   01                  LDA     #1
01860  50A6 B7   5107                STA     PDD
01870  50A9 20   04     50AF         BRA     PDM03
01880  50AB 4F                PDM02  CLRA
01890  50AC B7   5107                STA     PDD
01900  50AF B6   5106         PDM03  LDA     PD
01910  50B2 BB   5107                ADDA    PDD
01920  50B5 81   03                  CMPA    #3
01930  50B7 25   07     50C0         BCS     PDM04
01940  50B9 81   23                  CMPA    #35
01950  50BB 22   03     50C0         BHI     PDM04
01960  50BD B7   5106                STA     PD
01970  50C0 B6   5106         PDM04  LDA     PD
01980  50C3 80   03                  SUBA    #3
01990  50C5 C6   18                  LDB     #24
02000  50C7 BD   DA6D                JSR     LOCATE
02010  50CA 86   20                  LDA     #'        space
02020  50CC BD   D9D9                JSR     SOUT
02030  50CF 86   82                  LDA     #'_
02040  50D1 C6   05                  LDB     #5
02050  50D3 BD   D9D9         PDM05  JSR     SOUT
02060  50D6 5A                       DECB
02070  50D7 26   FA     50D3         BNE     PDM05
02080  50D9 86   20                  LDA     #'        space
02090  50DB 7E   D9D9                JMP     SOUT
02100                         *
02110  50DE BE   510A         WAIT   LDX     WW
02120  50E1 3D                WA01   MUL
```

```
02130   50E2 30    1F                   LEAX    -1,X
02140   50E4 26    FB    50E1           BNE     WA01
02150   50E6 39                         RTS
02160                              *
02170   5100                            ORG     $5100
02180   5100       01              DX   FCB     1
02190   5101       FF              DY   FCB     -1
02200   5102       0A              OLDX FCB     10
02210   5103       0A              OLDY FCB     10
02220   5104       0A              XX   FCB     10
02230   5105       0A              YY   FCB     10
02240   5106       14              PD   FCB     20
02250   5107       00              PDD  FCB     0
02260   5108       0000            SC   FDB     0
02270   510A       0001            WW   FDB     1
02280              5000                 END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=510B
PROGRAM ENTRY ADDR=5000
```

SCORE= 250

$D9A7：プリンタ・オープン

機　能　プリンタを論理番号なしでオープンします．

パラメータ　$2E7←オプション指定（S＝$53 or W＝$57）

$2E8←オプション指定（F＝$46 or N＝$4E）

オプション指定の内容については「ユーザーズマニュアルシステム解説　14－9」を御参照下さい．

アドレス　$D9A7

レジスタ　CC，A，B，X，Y

解　説　このルーチンはプリンタを論理番号なしでオープンすることができます．しかし，出力の際は下記の2つのルーチン以外は使用不可能です．

$D617：プリンタ・1文字出力

機　能　プリンタに1文字出力します．

パラメータ　Xレジスタ←（$5AA，$5AB）　（LDX$5AA）

Aレジスタ←出力文字コード

アドレス　$D617

レジスタ　CC，A，Y

解　説　このルーチンは，プリンタに1文字出力を行います．

単に1文字をプリンタに出力する場合，出力文字コードをAレジスタにセットして，$D659を呼ぶ（CCのみ破壊）方法もありますが，その場合，プリンタに対するBASICのPOS関数が正常に動作しません．

＄Ｄ６４Ａ：プリンタ改行

機　能　プリンタに対して，改行を行います．

アドレス　＄Ｄ６４Ａ

レジスタ　ＣＣ，Ａ，Ｂ，Ｙ，Ｕ

解　説　プリンタに対して，改行を行います．

具体的には，プリンタに，＄０Ｄ，＄０Ａを出力しますが，このとき＄１Ｅ３の内容により，片方のみ出力される場合があります．＄１Ｅ３にセットされているコードの文字は，「プリンタ改行」による改行のときには，プリンタへは送られずに無視します(たとえば，＄０Ｄがセットされていれば，＄０Ａのみ出力されることになります)．

サンプル

```
PAGE  001 (        ,        )

01000                           *
01010                           * printer output
01020                           *
01030               D9A7        PROPEN EQU    $D9A7   printer open
01040               D617        PROUT  EQU    $D617   printer output
01050               D64A        PRCRLF EQU    $D64A   printer CRLF
01060                                  OPT    M,NOG,NOS,P=255
01070  5000                            ORG    $5000
01080               5000        START  EQU    *
01090                           *
01100                           * printer open
01110                           *
01120  5000 86      53                 LDA    #'S
01130  5002 B7      02E7               STA    $2E7
01140  5005 86      4E                 LDA    #'N
01150  5007 B7      02E8               STA    $2E8
01160  500A BD      D9A7               JSR    PROPEN
01170                           *
01180                           * print output
01190                           *
01200  500D BD      D64A               JSR    PRCRLF
01210  5010 86      20                 LDA    #'      space
01220  5012 BE      05AA               LDX    $5AA
01230  5015 BD      D617               JSR    PROUT
01240  5018 4F                         CLRA
01250  5019 34      02          L01    PSHS   A
01260  501B 8D      41    505E         BSR    HEXOUT
01270  501D 86      20                 LDA    #'      space
01280  501F BD      D617               JSR    PROUT
01290  5022 35      02                 PULS   A
01300  5024 4C                         INCA
01310  5025 81      10                 CMPA   #$10
01320  5027 26      F0    5019         BNE    L01
01330  5029 BD      D64A               JSR    PRCRLF
01340  502C BD      D64A               JSR    PRCRLF
01350  502F 4F                         CLRA
01360  5030 34      02          L02    PSHS   A
01370  5032 8D      2A    505E         BSR    HEXOUT
01380  5034 C6      10                 LDB    #$10
01390  5036 A6      E4                 LDA    ,S
01400  5038 34      02          L03    PSHS   A
```

```
01410  503A 81    7F                  CMPA    #$7F     skip DEL
01420  503C 27    04    5042          BEQ     L04
01430  503E 81    20                  CMPA    #'       skip contorol code
01440  5040 24    02    5044          BCC     L05
01450  5042 86    20          L04     LDA     #'       space
01460  5044 BD    D617        L05     JSR     PROUT
01470  5047 86    20                  LDA     #'       space
01480  5049 BD    D617                JSR     PROUT
01490  504C 35    02                  PULS    A
01500  504E 8B    10                  ADDA    #$10
01510  5050 5A                        DECB
01520  5051 26    E5    5038          BNE     L03
01530  5053 BD    D64A                JSR     PRCRLF
01540  5056 35    02                  PULS    A
01550  5058 4C                        INCA
01560  5059 81    10                  CMPA    #$10
01570  505B 26    D3    5030          BNE     L02
01580  505D 39                        RTS
01590                         *
01600                         * subroutine:output Areg in HEX
01610                         *
01620  505E 81    0A          HEXOUT  CMPA    #$0A
01630  5060 25    02    5064          BCS     H01
01640  5062 8B    07                  ADDA    #$07
01650  5064 8B    30          H01     ADDA    #$30
01660  5066 7E    D617                JMP     PROUT
01670                         *
01680             5000                END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5068
PROGRAM ENTRY ADDR=5000

exec &H5000
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0	@	P	`	p	▁	┴		ー	タ	ミ	═	×
1			!	1	A	Q	a	q	▂	┬	。	ア	チ	ム	╞	円
2			"	2	B	R	b	r	▃	┤	「	イ	ツ	メ	╪	年
3			#	3	C	S	c	s	▄	├	」	ウ	テ	モ	╡	月
4			$	4	D	T	d	t	▅	￣	、	エ	ト	ヤ	◢	日
5			%	5	E	U	e	u	▆	─	・	オ	ナ	ユ	◣	時
6			&	6	F	V	f	v	▇	│	ヲ	カ	ニ	ヨ	◥	分
7			'	7	G	W	g	w	█	▕	ァ	キ	ヌ	ラ	◤	秒
8			(	8	H	X	h	x	▏	┌	ィ	ク	ネ	リ	♠	〒
9			)	9	I	Y	i	y	▎	┐	ゥ	ケ	ノ	ル	♥	市
A			*	:	J	Z	j	z	▍	└	ェ	コ	ハ	レ	♦	区
B			+	;	K	[	k	{	▌	┘	ォ	サ	ヒ	ロ	♣	町
C			,	<	L	¥	l	¦	▋	╭	ャ	シ	フ	ワ	●	村
D			-	=	M	]	m	}	▊	╮	ュ	ス	ヘ	ン	○	人
E			.	>	N	^	n	~	▉	╰	ョ	セ	ホ	゛	╱	▒
F			/	?	O	_	o		┼	╯	ッ	ソ	マ	゜	╲	

＄Ｅ２Ｅ９：２４時間計セット

機　能　２４時間計を指定時刻にセットします．

パラメータ　Ｕレジスタ←パラメータ先頭アドレス

Ｕレジスタ相対値

０：時（１０のケタ，０～２）
１：時（１のケタ，０～９）
２：分（１０のケタ，０～５）
３：分（１のケタ，０～９）
４：秒（１０のケタ，０～５）
５：秒（１のケタ，０～９）

アドレス　＄Ｅ２Ｅ９

レジスタ　ＣＣ，Ａ，Ｂ，Ｘ，Ｕ

解　説　２４時間計をＵレジスタが示すアドレスから，６バイトにあるパラメータにしたがってセットします．

＄Ｅ３５０：２４時間計読み出し

機　能　２４時間計の示している時刻を読み出します．

パラメータ　Ｕレジスタ←復帰情報エリア先頭アドレス

アドレス　＄Ｅ３５０　（＄Ｅ３４ＤにするとＵレジスタは＄２Ｆ４にセットされ，＄２Ｆ４～＄２Ｆ９に読み出します）

レジスタ　ＣＣ，Ａ，Ｂ，Ｘ

復帰情報　上項（＄Ｅ２Ｅ９：２４時間計セット）のパラメータと同じです．

解　説　２４時間計を示している時刻を読み出し，Ｕレジスタからの６バイトにセットします．

サンプル

```
PAGE  001 (        ,        )

01000                            *
01010                            * time set & display
01020                            *
01030                                    OPT     M,NOS,NOG,P=255
01040                E2E9        SETIME  EQU     $E2E9    set time
01050                E350        GETIME  EQU     $E350    get time
01060                D072        IN      EQU     $D072
01070                DB6D        INKEY   EQU     $DB6D    keyinput without wait
01080                D08E        OUT     EQU     $D08E
01090                9BDB        MESSAG  EQU     $9BDB
01100  5000                              ORG     $5000
01110                5000        START   EQU     *
01120                            *
01130                            * time set
01140                            *
01150  5000 30       8D 0060     SET     LEAX    MES1-1,PCR
01160  5004 BD       9BDB                JSR     MESSAG
01170  5007 33       8D 0083             LEAU    TIME,PCR
01180  500B 5F                           CLRB
01190  500C BD       D072        S01     JSR     IN
01200  500F 81       30                  CMPA    #'0
01210  5011 25       ED   5000           BCS     SET
01220  5013 81       39                  CMPA    #'9
01230  5015 22       E9   5000           BHI     SET
01240  5017 BD       D08E                JSR     OUT
01250  501A 80       30                  SUBA    #'0
01260  501C A7       C5                  STA     B,U
01270  501E 5C                           INCB
01280  501F C1       06                  CMPB    #6
01290  5021 27       0B   502E           BEQ     S02
01300  5023 C5       01                  BITB    #$01
01310  5025 26       E5   500C           BNE     S01
01320  5027 86       3A                  LDA     #':
01330  5029 BD       D08E                JSR     OUT
01340  502C 20       DE   500C           BRA     S01
01350  502E BD       E2E9        S02     JSR     SETIME   set time
01360                            *
01370                            * time display
01380                            *
01390  5031 30       8D 0054     DISP    LEAX    MES2-1,PCR
01400  5035 BD       9BDB                JSR     MESSAG
01410  5038 33       8D 0052             LEAU    TIME,PCR
01420  503C BD       E350                JSR     GETIME   get time
01430  503F 5F                           CLRB
01440  5040 A6       C5          D01     LDA     B,U
01450  5042 8B       30                  ADDA    #'0
01460  5044 BD       D08E                JSR     OUT
01470  5047 5C                           INCB
01480  5048 A6       C5                  LDA     B,U
01490  504A 8B       30                  ADDA    #'0
01500  504C BD       D08E                JSR     OUT
01510  504F 5C                           INCB
01520  5050 C1       06                  CMPB    #6
01530  5052 27       07   505B           BEQ     D02
01540  5054 86       3A                  LDA     #':
01550  5056 BD       D08E                JSR     OUT
01560  5059 20       E5   5040           BRA     D01
01570  505B BD       DB6D        D02     JSR     INKEY    keyinput
01580  505E 27       D1   5031           BEQ     DISP     if no keyin
01590  5060 81       20                  CMPA    #'       keyin=space ?
01600  5062 27       9C   5000           BEQ     SET      yes,set time again
01610  5064 39                           RTS              no,return
01620                            *
01630                            * messages
01640                            *
01650  5065          0C          MES1    FCB     $0C,$12,5,5 cls:locate 5,5
01660  5069          20                  FCC     ' PRESENT TIME = 時時:分分:秒秒'
01670  5081          1D                  FCB     $1D,$1D,$1D,$1D,$1D,$1D,$1D,$1D
01680                            *                 chr$($1d)='<-' left cursor
```

```
01690  5089     00              FCB    0
01700  508A     12       MES2   FCB    $12,21,5 locate 21,5
01710  508D     00              FCB    0
01720                    *
01730  508E     0006     TIME   RMB    6
01740                    *
01750           5000            END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5093
PROGRAM ENTRY ADDR=5000


exec &H5000




       PRESENT TIME = 22:48:04
Ready
```

$EBE5：PSGイニシャライズ

機　　能　ＰＳＧをイニシャライズします．

アドレス　$ＥＢＥ５

レジスタ　ＣＣ

解　　説　ＰＳＧの各レジスタを初期化します．具体的には，レジスタ０７に$Ｆ８を，その他のレジスタに０をセットします．

$EBCF：PSGレジスタ・セット

機　　能　指定したＰＳＧのレジスタに，値をセットします．

パラメータ　Ａレジスタ←レジスタ番号（０～１３）
Ｂレジスタ←値

アドレス　$ＥＢＣＦ

レジスタ　ＣＣ

解　　説　指定したＰＳＧのレジスタに，値をセットします．各レジスタの機能等については，「ユーザーズマニュアル」を御参照下さい．具体的には，$ＦＤ０Ｄ，$ＦＤ０Ｅを操作してレジスタを選択し，データを書き込みます．

サンプル

```
PAGE  001 (        ,        )

01000                          *
01010                          * PSG control program
01020                          *
01030                                  OPT     M,NOS,NOG,P=255
01040                  EBE5    PSGINT  EQU     $EBE5
01050                  EBCF    PSGSET  EQU     $EBCF
01060                  D08E    OUT     EQU     $D08E
01070                  D072    IN      EQU     $D072
01080                  AC3D    A16OUT  EQU     $AC3D
01081                  9BDB    MESSAG  EQU     $9BDB
01083                  9B50    CRLF    EQU     $9B50
01090   5000                           ORG     $5000
01100                  5000    START   EQU     *
01110                          *
01111                          * initialize
01112                          *
01120   5000 BD        EBE5            JSR     PSGINT
01130   5003 30        8D 00FF         LEAX    MES1-1,PCR
01140   5007 BD        9BDB            JSR     MESSAG
01150   500A C6        0D              LDB     #13
01160   500C 33        8D 0159         LEAU    REG,PCR
01170   5010 6F        C5      I01     CLR     B,U
```

```
01180  5012 5A                       DECB
01190  5013 2A    FB   5010          BPL     IO1
01200  5015 86    F8                 LDA     #$F8
01210  5017 A7    47                 STA     7,U
01220                          *
01230                          * display register value
01240                          *
01250  5019 30    8D 00FF  DISP      LEAX    MES2-1,PCR
01260  501D BD    9BDB               JSR     MESSAG
01270  5020 33    8D 0145            LEAU    REG,PCR
01280  5024 5F                       CLRB
01290  5025 34    04       DO1       PSHS    B
01300  5027 86    52                 LDA     #'R
01310  5029 BD    D08E               JSR     OUT
01320  502C 86    30                 LDA     #'0
01330  502E C1    0A                 CMPB    #10
01340  5030 25    04   5036          BCS     DO2
01350  5032 86    31                 LDA     #'1
01360  5034 C0    0A                 SUBB    #10
01370  5036 BD    D08E     DO2       JSR     OUT
01380  5039 CB    30                 ADDB    #'0
01390  503B 1F    98                 TFR     B,A
01400  503D BD    D08E               JSR     OUT
01410  5040 86    3D                 LDA     #'=
01420  5042 BD    D08E               JSR     OUT
01430  5045 86    24                 LDA     #'$
01440  5047 BD    D08E               JSR     OUT
01450  504A E6    E4                 LDB     ,S
01460  504C A6    C5                 LDA     B,U
01470  504E BD    AC3D               JSR     A16OUT
01471  5051 BD    9B50               JSR     CRLF
01480  5054 35    04                 PULS    B
01490  5056 5C                       INCB
01500  5057 C1    0E                 CMPB    #14
01510  5059 26    CA   5025          BNE     DO1
01520                          *
01530                          * set register
01540                          *
01550  505B 30    8D 00D4  SET       LEAX    MES3-1,PCR
01560  505F BD    9BDB               JSR     MESSAG
01570  5062 BD    D072               JSR     IN
01580  5065 81    0D                 CMPA    #$0D    if CR then end of prog.
01590  5067 26    04   506D          BNE     SO1
01595  5069 BD    EBE5               JSR     PSGINT  sound stop
01600  506C 39                       RTS
01610  506D BD    D08E     SO1       JSR     OUT     get register number
01620  5070 80    30                 SUBA    #'0
01630  5072 C6    0A                 LDB     #10
01640  5074 3D                       MUL
01645  5075 34    04                 PSHS    B
01650  5077 BD    D072               JSR     IN
01660  507A BD    D08E               JSR     OUT
01670  507D 80    30                 SUBA    #'0
01680  507F AB    E4                 ADDA    ,S
01690  5081 A7    E4                 STA     ,S
01691  5083 81    0E                 CMPA    #14
01692  5085 25    04   508B          BCS     SO2
01693  5087 35    02                 PULS    A
01694  5089 20    D0   505B          BRA     SET
01700  508B 30    8D 00BF  SO2       LEAX    MES4-1,PCR get value
01710  508F BD    9BDB               JSR     MESSAG
01720  5092 8D    0E   50A2          BSR     INPUT
01730  5094 35    02                 PULS    A
01740  5096 33    8D 00CF            LEAU    REG,PCR set register
01750  509A E7    C6                 STB     A,U
01760  509C BD    EBCF               JSR     PSGSET
01770  509F 16    FF77 5019          LBRA    DISP
01780                          *
01790                          * subroutine:input hex or binary in Breg
01800                          *
01810  50A2 BD    D072     INPUT     JSR     IN      input '$' or '%'
01820  50A5 81    24                 CMPA    #'$
01830  50A7 27    25   50CE          BEQ     HEX
01840  50A9 81    25                 CMPA    #'%
```

```
01850   50AB 26    F5    50A2         BNE     INPUT
01860   50AD BD    D08E          BIN    JSR.    OUT        if binary
01870   50B0 5F                         CLRB
01880   50B1 34    04                   PSHS    B
01890   50B3 BD    D072          B01    JSR     IN
01900   50B6 81    31                   CMPA    #'1
01910   50B8 27    0A    50C4           BEQ     B02
01920   50BA 81    30                   CMPA    #'0
01930   50BC 27    06    50C4           BEQ     B02
01940   50BE 81    0D                   CMPA    #$0D
01950   50C0 26    F1    50B3           BNE     B01
01960   50C2 35    84                   PULS    B,PC
01965   50C4 BD    D08E          B02    JSR     OUT
01970   50C7 80    30                   SUBA    #'0
01980   50C9 46                         RORA
01990   50CA 69    E4                   ROL     ,S
02000   50CC 20    E5    50B3           BRA     B01
02010                            *
02020   50CE BD    D08E          HEX    JSR     OUT        if hex
02030   50D1 5F                         CLRB
02040   50D2 34    04                   PSHS    B
02050   50D4 BD    D072          H01    JSR     IN
02060   50D7 81    0D                   CMPA    #$0D
02070   50D9 26    02    50DD           BNE     H02
02080   50DB 35    84                   PULS    B,PC
02090   50DD 81    30            H02    CMPA    #'0
02100   50DF 25    F3    50D4           BCS     H01
02110   50E1 81    39                   CMPA    #'9
02120   50E3 23    0F    50F4           BLS     H03
02130   50E5 81    41                   CMPA    #'A
02140   50E7 25    EB    50D4           BCS     H01
02150   50E9 81    46                   CMPA    #'F
02160   50EB 22    E7    50D4           BHI     H01
02170   50ED BD    D08E                 JSR     OUT
02180   50F0 80    37                   SUBA    #'A-10
02190   50F2 20    05    50F9           BRA     H04
02200   50F4 BD    D08E          H03    JSR     OUT
02210   50F7 80    30                   SUBA    #'0
02220   50F9 68    E4            H04    ASL     ,S
02230   50FB 68    E4                   ASL     ,S
02240   50FD 68    E4                   ASL     ,S
02250   50FF 68    E4                   ASL     ,S
02260   5101 AB    E4                   ADDA    ,S
02270   5103 A7    E4                   STA     ,S
02280   5105 20    CD    50D4           BRA     H01
02290                            *
02300                            * messages
02310                            *
02320   5107       0D0A          MES1   FDB     $0D0A
02330   5109       2A                   FCC     '* PSG initialized *
02340   511C       00                   FCB     0
02350   511D       0D0A          MES2   FDB     $0D0A
02360   511F       2A                   FCC     '* REGISTER VALUE *'
02370   5131       0D0A                 FDB     $0D0A
02380   5133       00                   FCB     0
02390   5134       0D0A          MES3   FDB     $0D0A
02400   5136       20                   FCC     '        REGISTER NUMBER =R'
02410   514E       00                   FCB     0
02420   514F       0D0A          MES4   FDB     $0D0A
02430   5151       20                   FCC     ' VALUE ($HH or %B..B) ='
02440   5168       00                   FCB     0
02450                            *
02460   5169       000E          REG    RMB     14         register save area
02470                            *
02480              5000                 END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5176
PROGRAM ENTRY ADDR=5000
```

$DB3F:BEEP ON

$DB44:BEEP OFF

機　　能	BEEP音をコントロールします.
アドレス	$DB3F:BEEP音を鳴らします.(連続)BEEP1に相当. $DB44:BEEP音を止める.BEEP0に相当.
レジスタ	CC, X

解　　説　BIOSを使用して, BEEP音をコントロールします.

$DB38:BEEP

機　　能	BEEP音を一定時間鳴らします.
アドレス	$DB38
レジスタ	CC, A

解　　説　BASICのBEEPに相当する動作をします. 具体的には, ファイル番号0で, $07 (BELL) を出力します.

$D778:MOTOR ON

$D77B:MOTOR OFF

機　　能	カセットテープのモータをコントロールします.
アドレス	$D778:MOTOR ON $D77B:MOTOR OFF
レジスタ	CC, A

解　　説　BIOSを使用して, モータをコントロールします.

$EB77：マルチページ機能設定

機　能	マルチページレジスタに値を設定します.
パラメータ	Aレジスタ←アクティブVRAMコード（0～7）
	Bレジスタ←ディスプレイVRAMコード（0～7）

VRAMコード：bit　0：B　指定する画面のビットを
　　　　　　　　　1：R　　'1'にする.
　　　　　　　　　2：G

アドレス	$EB77
レジスタ	CC, A, B

解　説　マルチページレジスタに値を設定して，マルチページを実現します．パラメータを設定するにあたり注意する点は，指定する画面のビットを'1'とすることです．したがって$FD37に書き込む場合とビットの設定が逆（$FD37に書き込む場合には，指定する画面のビットを'0'にする）となります．

また，このルーチンは，サブCPUのコマンド動作が終了（サブCPUがレディになる）を待って，マルチページレジスタに値を設定します．

サンプル

```
PAGE  001 (        ,        )

01000                          *
01010                          * set active VRAM & display VRAM
01020                          *
01030                                   OPT     M,NOS,NOG,P=255
01040               EB77       SCREEN   EQU     $EB77    set vram flag (SCREEN)
01050               D08E       OUT      EQU     $D08E
01060               D072       IN       EQU     $D072
01070               9BDB       MESSAG   EQU     $9BDB
01080   5000                            ORG     $5000
01090               5000       START    EQU     *
01100                          *
01110   5000 30     8D 0032             LEAX    MES1-1,PCR set active VRAM
01120   5004 BD     9BDB                JSR     MESSAG
01130   5007 8D     11    501A          BSR     INPUT
01140   5009 34     04                  PSHS    B
01150   500B 30     8D 003F             LEAX    MES2-1,PCR set display VRAM
01160   500F BD     9BDB                JSR     MESSAG
01170   5012 8D     06    501A          BSR     INPUT
01180   5014 35     02                  PULS    A
01190   5016 BD     EB77                JSR     SCREEN  set VRAM flag
01200   5019 39                         RTS
01210                          *
01220                          * subroutine:input binary in Breg
01230                          *
01240   501A 5F                INPUT    CLRB
01250   501B 34     04                  PSHS    B
01260   501D BD     D072       I01      JSR     IN
01270   5020 81     30                  CMPA    #'0
```

```
01280  5022 27   04   5028          BEQ    I02
01290  5024 81   31                 CMPA   #'1
01300  5026 26   F5   501D          BNE    I01
01310  5028 BD   D08E       I02     JSR    OUT
01320  502B 80   30                 SUBA   #'0
01330  502D 46                      RORA
01340  502E 69   E4                 ROL    ,S
01350  5030 5C                      INCB
01360  5031 C1   03                 CMPB   #3
01370  5033 26   E8   501D          BNE    I01
01380  5035 35   84                 PULS   B,PC
01390                       *
01400  5037      0D0A       MES1    FDB    $0D0A
01410  5039      20                 FCC    ' active VRAM = GRB'
01420  504B      1D                 FCB    $1D,$1D,$1D
01430  504E      00                 FCB    0
01440  504F      0D0A       MES2    FDB    $0D0A
01450  5051      64                 FCC    'display VRAM = GRB'
01460  5063      1D                 FCB    $1D,$1D,$1D
01470  5066      00                 FCB    0
01480                       *
01490            5000               END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5066
PROGRAM ENTRY ADDR=5000


exec &H5000

 active VRAM = 100
display VRAM = 100
Ready
```

＄C８０A：パレットレジスタ機能

機　能　パレットレジスタに値を設定します．

パラメータ　Bレジスタ←パレットコード（１～７）
　　　　　　　Aレジスタ←カラーコード（０～７）

アドレス　＄C８０A

レジスタ　CC，A

解　説　このルーチンを使用することにより，カラーパレットを任意の色に割り付けることができます．BASICのCOLOR＝（B，A）に相当します．

具体的には，パレットコードによりI／Oアドレスを決定し，カラーコードを書き込みます．

パレットレジスタにおいては，パレットコード０にも任意の色を設定できるのですが，このルーチンでは，パレットコード０には常にカラーコード０（黒）を設定します．パレットコード０に黒以外の色を設定したい場合には，＄FD３８にカラーコードを書き込みます．これにより，疑似的にボーダー（帰線区間）カラーの設定ができます．

サンプル

```
PAGE  001 (          ,          )

01000                           *
01010                           * set color palette
01020                           *
01030                                  OPT    M,NOS,NOG,P=255
01040              C80A         PALET  EQU    $C80A
01050              D072         IN     EQU    $D072
01060              D08E         OUT    EQU    $D08E
01070              9BDB         MESSAG EQU    $9BDB
01080      5000                        ORG    $5000
01090              5000         START  EQU    *
01100                           *
01110      5000 30 8D 0036             LEAX   MES-1,PCR get palette code
01120      5004 BD 9BDB                JSR    MESSAG
01130      5007 BD D072         C01    JSR    IN
01140      500A 81 30                  CMPA   #'0
01150      500C 25 F9   5007           BCS    C01
01160      500E 81 37                  CMPA   #'7
01170      5010 22 F5   5007           BHI    C01
01180      5012 BD D08E                JSR    OUT
01190      5015 80 30                  SUBA   #'0
01200      5017 34 02                  PSHS   A
01210      5019 86 2C                  LDA    #',
01220      501B BD D08E                JSR    OUT
01230      501E BD D072         C02    JSR    IN        get color code
01240      5021 81 30                  CMPA   #'0
01250      5023 25 F9   501E           BCS    C02
01260      5025 81 37                  CMPA   #'7
01270      5027 22 F5   501E           BHI    C02
01280      5029 BD D08E                JSR    OUT
01290      502C 80 30                  SUBA   #'0
```

```
01300  502E 34   02               PSHS   A
01310  5030 86   29               LDA    #')
01320  5032 BD   D08E             JSR    OUT
01330  5035 35   06               PULS   A,B
01340  5037 BD   C80A             JSR    PALET   set palette
01350  503A 39                    RTS
01360                        *
01370  503B      0D0A        MES  FDB    $0D0A
01380  503D      73               FCC    'set palette  :  COLOR=('
01390  5054      00               FCB    0
01400                        *
01410            5000             END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5054
PROGRAM ENTRY ADDR=5000


exec &H5000

set palette  :  COLOR=(4,7)
Ready
```

＄C495：大文字変換

機　能　英小文字を英大文字に変換します.

パラメータ　Aレジスタ←文字コード（任意）

アドレス　＄C495

レジスタ　CC，A

復帰情報　Aレジスタ：変換された文字コード

解　説　パラメータで与えられた文字が，英小文字であれば，英大文字に変換します．英小文字以外の場合には，この変換はされません．

＄94FD，＄9506：文字チェック

機　能　与えられた文字が，指定された種類の文字かどうか判別します.

パラメータ　Aレジスタ←文字コード（任意）

アドレス　＄94FD：英大文字であるかを判別

＄9506：数字（0～9）であるかを判別

レジスタ　CC

復帰情報　CCレジスタ・キャリーフラグ（bit 0）

指定された種類の文字だった場合…………1

指定された種類の文字でなかった場合……0

サンプル

```
PAGE  001 (        ,        )

01000                         *
01010                         * character check
01020                         *
01030                                 OPT    M,NOS,NOG,P=255
01040              D072       IN      EQU    $D072
01050              D08E       OUT     EQU    $D08E
01060              9BDB       MESSAG  EQU    $9BDB
01070              94FD       ALPHA   EQU    $94FD
01080              9506       NUMBER  EQU    $9506
01090              C495       CAPS    EQU    $C495
01100  5000                           ORG    $5000
01110              5000       START   EQU    *
01120                         *
01130  5000 30     8D 004E    LOOP    LEAX   MES1-1,PCR
01140  5004 BD     9BDB               JSR    MESSAG
01150  5007 BD     D072       L01     JSR    IN
01160  500A 81     0D                 CMPA   #$0D      if CR then end
01170  500C 27     44    5052         BEQ    L05
01180  500E 81     20                 CMPA   #'        control code skip
```

```
01190  5010 25   F5   5007          BCS    L01
01200  5012 81   7F                 CMPA   #$7F     DEL skip
01210  5014 27   F1   5007          BEQ    L01
01220  5016 BD   D0BE               JSR    OUT      echo back
01230  5019 BD   C495               JSR    CAPS     capital
01240  501C 34   02                 PSHS   A
01250  501E 30   8D 0049            LEAX   MES2-1,PCR
01260  5022 BD   9BDB               JSR    MESSAG
01270  5025 A6   E4                 LDA    ,S
01280  5027 BD   D0BE               JSR    OUT
01290  502A 30   8D 004C            LEAX   MES3-1,PCR
01300  502E BD   9BDB               JSR    MESSAG
01310  5031 35   02                 PULS   A
01320  5033 BD   94FD               JSR    ALPHA    alphabet ?
01330  5036 25   06   503E          BCS    L02
01340  5038 30   8D 0044            LEAX   MESA-1,PCR yes,alphabet
01350  503C 20   0F   504D          BRA    L04
01360  503E BD   9506       L02     JSR    NUMBER   number ?
01370  5041 25   06   5049          BCS    L03
01380  5043 30   8D 0044            LEAX   MESN-1,PCR yes,number
01390  5047 20   04   504D          BRA    L04
01400  5049 30   8D 004D    L03     LEAX   MESX-1,PCR else
01410  504D BD   9BDB       L04     JSR    MESSAG
01420  5050 20   AE   5000          BRA    LOOP     next
01430  5052 39              L05     RTS             end of program
01440                       *
01450                       * messages
01460                       *
01470  5053      0D0A       MES1    FDB    $0D0A
01480  5055      20                 FCC    ' input one character ='
01490  506B      00                 FCB    0
01500  506C      0D0A       MES2    FDB    $0D0A
01510  506E      20                 FCC    / character '/
01520  507A      00                 FCB    0
01530  507B      27         MES3    FCC    /' is /
01540  5080      00                 FCB    0
01550  5081      41         MESA    FCC    'ALPHABET. '
01560  508B      00                 FCB    0
01570  508C      4E         MESN    FCC    'NUMBER (0..9).'
01580  509A      00                 FCB    0
01590  509B      6E         MESX    FCC    'neither ALPHABET nor NUMBER.'
01600  50B7      00                 FCB    0
01610                       *
01620            5000               END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50B7
PROGRAM ENTRY ADDR=5000


exec &H5000

 input one character =5
 character '5' is NUMBER (0..9).
 input one character =Q
 character 'Q' is ALPHABET.
 input one character =q
 character 'Q' is ALPHABET.
 input one character =e
 character 'E' is ALPHABET.
 input one character =@
 character '@' is neither ALPHABET nor NUMBER.
 input one character =
Ready
```

4. System Subroutines 2

本章は，ＢＡＳＩＣのＲＯＭ上に存在する各種のルーチンのうち，数値演算関係のものでユーザーの使用頻度の多いものを解説します．

本章では，各ルーチンを用途別に分類し，同種のものはなるべくまとめて解説しています．なお，本章で示されるルーチンは，Ｆ－ＢＡＳＩＣのワークエリアを破壊するものが多いので，利用する際には，十分な考慮が必要です．また，前述した一行出力（１）（＄９ＢＤＢ）のルーチンは，ＦＡＣ１（後述）を破壊しますので注意して下さい．

◀各項目のみかた▶

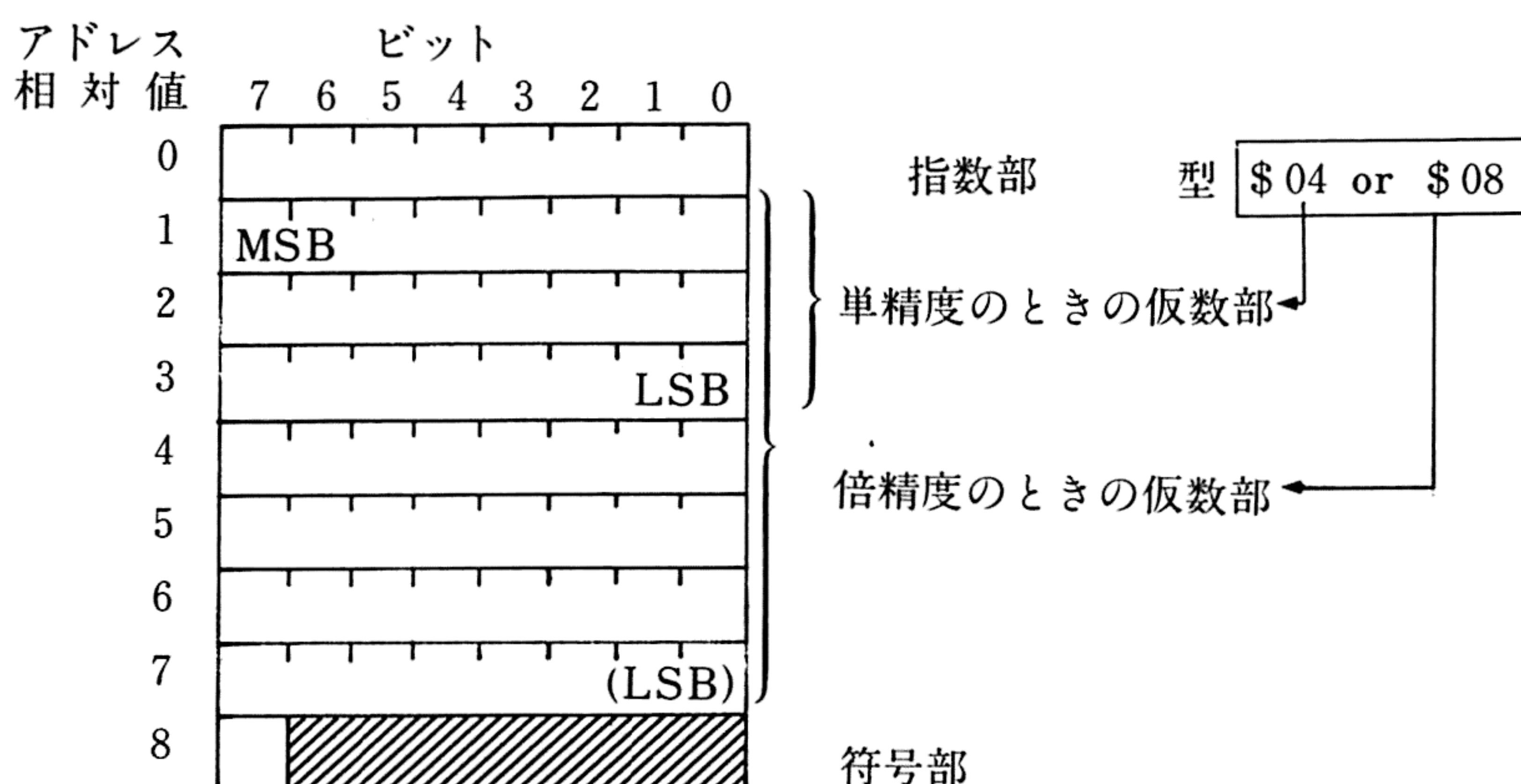
レジスタ，解説等は，これまでの章と同じです．ただし サンプル のマシン語プログラムは，これまでの章と違い，一部を除いてポジションインディペンデント（位置独立）ではありません．これは，プログラムをなるべく複雑にしないようにするための処置です．

また，解説では触れていませんが，各ルーチンにおいて何らかのエラー（０で割る，Type Mismach 等）が生じた場合には，ＢＡＳＩＣのエラー処理を行い，ＢＡＳＩＣのコマンドレベルへ制御が移りますので注意して下さい．

ＦＡＣの構造

Ｆ－ＢＡＳＩＣインタプリンタは，ＣＰＵである６８０９によって実行されているわけですが，６８０９は浮動小数の演算機能を持っていません．そのため，Ｆ－ＢＡＳＩＣでは，ＦＡＣ（フローティング・アキュムレータ）と呼ばれる仮想のレジスタをＲＡＭ上に用意し，このＦＡＣに対し各種の演算を施します．

このＦＡＣは，メモリ上に３つ確保されています．以下にその構造を示します．

◀実数の場合▶

	FACのアドレス	型のアドレス
FAC1	$74～$7C	$17
2	$82～$8A	$5D
3	$28～$30	——

指数部　実数を２進数の浮動小数点表現した場合の指数が入ります．

＄８０のゲタはかせにより値を表示しています（２の補数表現ではありません）．

＄０１………2^{-127}
　⋮
＄７Ｆ………2^{-1}
＄８０………2^{0}
＄８１………2^{1}
　⋮
＄ＦＦ………2^{+127}

［＄００は仮数部の値に関係なく，実数値０を示すものとして扱われます．］

仮数部　倍精度実数の場合は7バイト，単精度実数の場合は3バイトが使用されます．

仮数部は符号なし2進数で示されています．浮動小数点表現では，仮数部の最上位ビットは常に1となるので，アドレス相対値1の最上位ビットは常に1となります（ただし，ＦＡＣが0を示すときは例外です）．

符号部　相対値8の最上位ビットが仮数の符号を示します．0のとき正，1のとき負を示します．

指数部をｅ，仮数部の各ビットを△，符号部を±で示すと，実際の数値は，

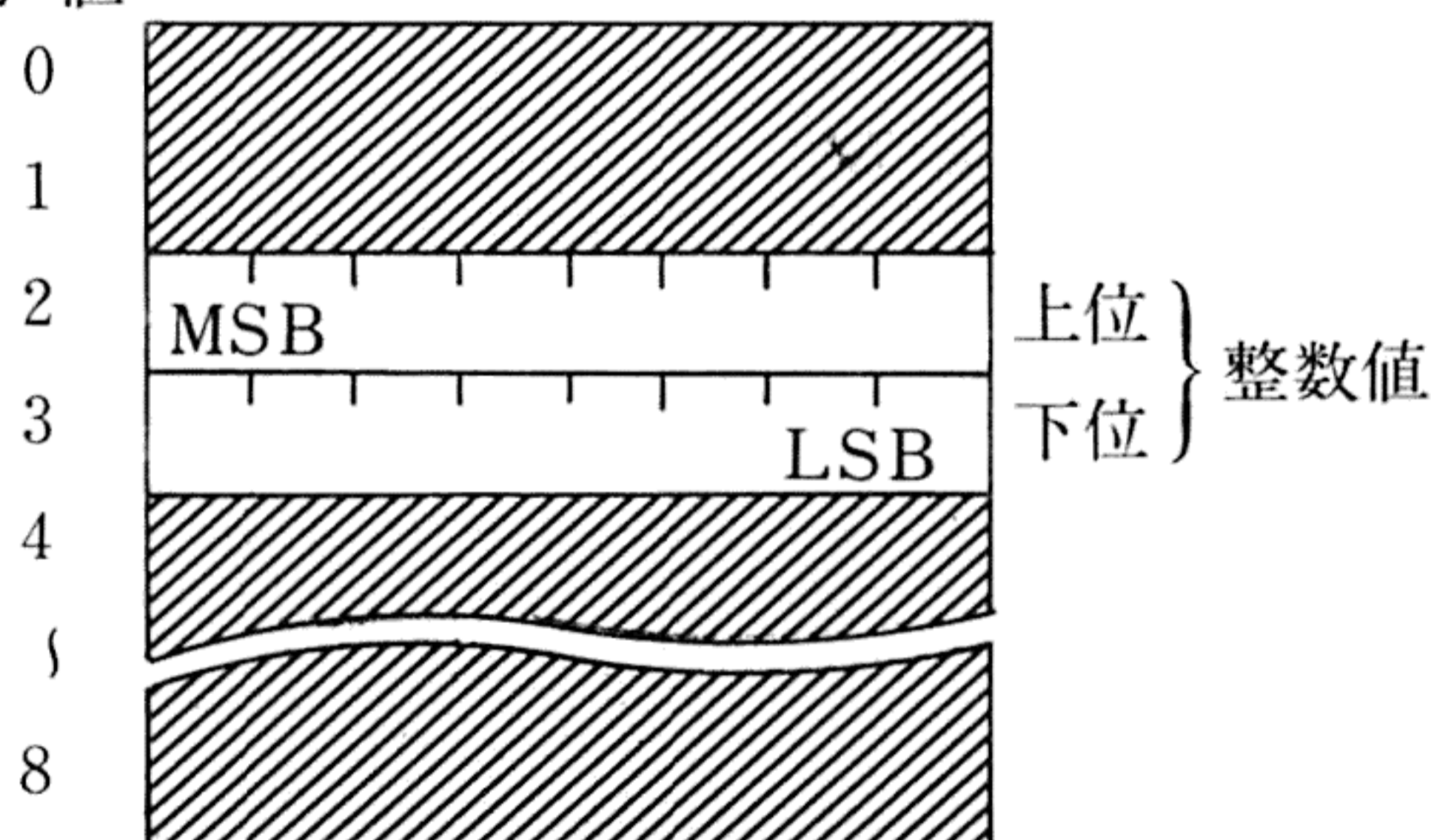
┌常に1．

X＝±．△△△△△……△△△X 2^e

となります．しかし，例外として，実数の0は，指数部を0として示します．

◀整数の場合▶

アドレス相対値	内容	
0	(斜線)	
1	(斜線)	
2	MSB	上位 }整数値
3	LSB	下位 }整数値
4	(斜線)	
∫		
8	(斜線)	

型　＄02

アドレスは実数の場合と同じです。

整数値　2の補数表示により示します．

サンプル 数値を入力すると，その数値の格納状態を示します．

```
PAGE  001 (        ,        )

01000                          *
01010                          * dump number
01020                          *
01030                                   OPT     M,NOS,NOG,P=255
01040               D807       LINEIN EQU     $D807     line input
01050               9BDB       MESSAG EQU     $9BDB     output from (X+1) till 0
01060               D08E       OUT    EQU     $D08E     output 1 character
01070               B50B       STRFAC EQU     $B50B     convert string into FAC1
01080               AC3D       A160UT EQU     $AC3D     output Areg in HEX
01090   5000                          ORG     $5000
01100               5000       START  EQU     *
01110                          *
01120   5000 9E     D9                LDX     $D9       save POINTER
01130   5002 BF     507E              STX     TEMP
01140   5005 8E     5063       NEXT   LDX     #PROMPT-1 output message
01150   5008 BD     9BDB              JSR     MESSAG
01160   500B BD     D807              JSR     LINEIN
01170   500E 6D     01                TST     1,X       if null then end of program
01180   5010 27     4C   505E         BEQ     END
01190   5012 34     10                PSHS    X
01200   5014 8E     5071              LDX     #MES-1    output message
01210   5017 BD     9BDB              JSR     MESSAG
01220   501A 35     10                PULS    X
01230   501C 9F     D9                STX     $D9       set pointer
01240   501E BD     B50B              JSR     STRFAC
01250                          *
01260   5021 5F                       CLRB              clear offset
01270   5022 CE     0074              LDU     #$74      FAC1 base address
01280   5025 96     17         LOOP   LDA     $17       load var.type
01290   5027 81     02                CMPA    #2
01300   5029 26     08   5033         BNE     L01
01310   502B C1     02                CMPB    #2        if INT,disp 2..3
01320   502D 25     1B   504A         BLO     L04
01330   502F C1     04                CMPB    #4
01340   5031 24     17   504A         BHS     L04
01350   5033 81     04         L01    CMPA    #4
01360   5035 26     08   503F         BNE     L02
01370   5037 C1     04                CMPB    #4        if SNG,disp 0..3,8
01380   5039 25     08   5043         BLO     L03
01390   503B C1     08                CMPB    #8
01400   503D 26     0B   504A         BNE     L04
01410   503F 81     03         L02    CMPA    #3
01420   5041 27     1B   505E         BEQ     END       if STR ,end
01430   5043 A6     C5         L03    LDA     B,U       load memory
01440   5045 BD     AC3D              JSR     A160UT    output
01450   5048 20     08   5052         BRA     L05
01460   504A 86     5F         L04    LDA     #'_       not disp memory
01470   504C BD     D08E              JSR     OUT
01480   504F BD     D08E              JSR     OUT
01490   5052 86     20         L05    LDA     #'        space
01500   5054 BD     D08E              JSR     OUT
01510   5057 5C                       INCB
01520   5058 C1     09                CMPB    #9
01530   505A 26     C9   5025         BNE     LOOP
01540   505C 20     A7   5005         BRA     NEXT
01550                          *
01560   505E BE     507E       END    LDX     TEMP
01570   5061 9F     D9                STX     $D9
01580   5063 39                       RTS
01590                          *
01600   5064        0D0A       PROMPT FDB     $0D0A
01610   5066        20                FCC     '    number ='
01620   5071        00                FCB     0
01630   5072        6F         MES    FCC     'on memory ='
01640   507D        00                FCB     0
01650                          *
01660   507E        0001       TEMP   FDB     1
01670                          *
01680               5000              END     START
TOTAL ERRORS 00000--00000
```

```
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=507F
PROGRAM ENTRY ADDR=5000


exec &H5000

    number =1
on memory =__ __ 00 01 __ __ __ __ __
    number =-32767
on memory =__ __ 80 01 __ __ __ __ __
    number =3.25
on memory =82 D0 00 00 __ __ __ __ 00
    number =-3.25
on memory =82 D0 00 00 __ __ __ __ FF
    number =.625#
on memory =80 A0 00 00 00 00 00 00 00
    number =-.625#
on memory =80 A0 00 00 00 00 00 00 FF
    number =.00001
on memory =70 A7 C5 AB __ __ __ __ 00
    number =

Ready
```

ＦＡＣと文字列との間の変換

数値を表わした文字列と数値（ＦＡＣ）との間の変換を示します.

◀文字列を数値へ▶

機　能　文字列を数値へ変換する.

パラメータ　＄Ｄ９←文字列先頭アドレス－１

＄D9
↓
| | 1 2 3…文字列…9 | ←関係のない文字 |

アドレス　倍精度で数値を得たい時：＄Ｂ５０６
それ以外の時　　　　　：＄Ｂ５０Ｂ

レジスタ　ＣＣ，Ａ，Ｂ，Ｘ，Ｕ

復帰情報　ＦＡＣ１（＄７４～＄７Ｃ）：得られた数値
＄１７：数値の型
＄Ｄ９：関係のない文字のアドレス

解　説　数値定数を表わす文字列を，適当な型の数値としてＦＡＣ１に格納します.

＄Ｂ５０６のとき，数値の型は以下の場合を除いて，倍精度型となります.

1）８進（&,&０），１６進（&Ｈ）のデータ（整数型となります）
2）末尾に！を伴なったデータ（単精度型となります）
3）Ｅを使った指数形式のデータ（単精度型となります）

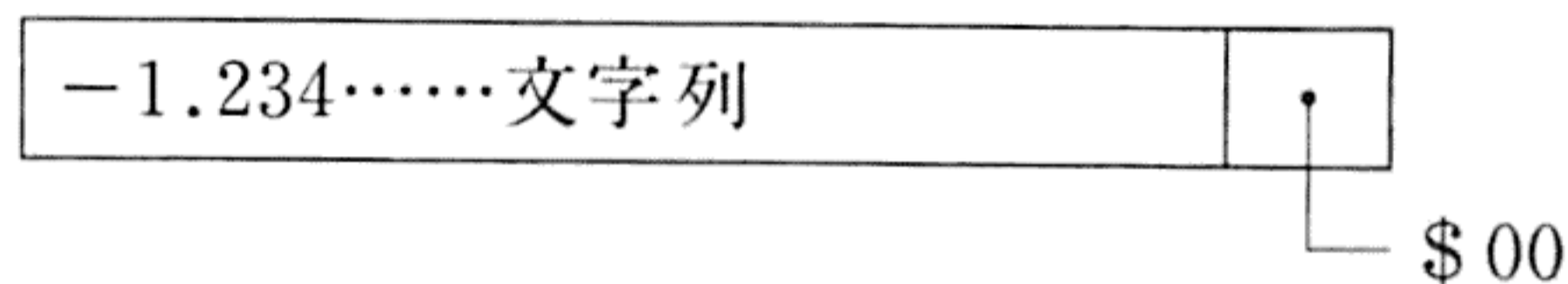
＄Ｂ５０Ｂのときには，適当な型が選ばれます（#，%，！によって指定があれば，それに従います）.

◀数値を文字列へ▶

機　能	数値を文字列へ変換する．
パラメータ	ＦＡＣ１←数値
	＄１７←数値の型
	＄Ｂ８～＄ＢＡ←フォーマット指定（＄Ｂ６２２のときのみ有効）
	フォーマット付　　＄Ｂ６２２
	フォーマットなし　＄Ｂ６２０
レジスタ	ＣＣ，Ａ，Ｂ，Ｘ，Ｕ
復帰情報	Ｘレジスタ：文字列先頭アドレス

X＝＄541
↓

−1.234……文字列	・

└ ＄00

解　説　数値の型にしたがってＦＡＣ１に格納されている数値を，数値を示す文字列に変換します．文字列は＄５４１からに格納され，Ｘレジスタがその先頭を示します．

＄Ｂ６２２の場合には，フォーマット指定にしたがって文字列に変換します．

◀フォーマット指定▶

＄B8——小数点以下のケタ数(小数点を含む)
＄B9——小数点より左(整数部分)のケタ数

例）␣１２３・４５６７
　　＄B9　　＄B8
　　└数値のケタ数の方が小さい場合には右詰めになる

＄ＢＡ　ビット７——フォーマット指定の可否．このビットが０の場合フォーマットは行わずフォーマット指定は無効となります．

ビット６——カンマ指定．このビットが１のとき，数値の整数部が３ケタ毎にカンマ（，）で区切られます．

ビット５——アスタリスク指定．このビットを１にした場合，数値桁数が満たない部分をアスタリスク（＊）で埋めます．

ビット４——エンマーク指定．このビットを１にすると，数値の直

前にエンマーク（¥）を出力します．

ビット3——プラス指定．このビットを1にすると，数値が負のときはマイナス符号（−），正のときはプラス符号（+）を数値の前に出力します．

ビット2——マイナス指定．このビットを1にすると，数値が負のときに，マイナス符号（−）を数値の直後に出力します．

ビット0——指数形式指定．このビットを1にすると，数値は指数形式で出力されます．

サンプル

```
PAGE  001 (          ,          )

01000                          *
01010                          * print using
01020                          *
01030                                  OPT    M,NOS,NOG,P=255
01040              D807        LINEIN EQU    $D807    line input
01050              9BDB        MESSAG EQU    $9BDB    output from (X+1) till 0
01060              D08E        OUT    EQU    $D08E    output 1 character
01070              D072        IN     EQU    $D072    input 1 character
01080              B50B        STRFAC EQU    $B50B    convert string into FAC1
01090              B622        FACSTR EQU    $B622    convert FAC1 into string
01100  5000                           ORG    $5000
01110              5000        START  EQU    *
01120                          *
01130  5000 9E     D9                 LDX    $D9      save POINTER
01140  5002 BF     513D               STX    TEMP
01150  5005 8E     50AD        NEXT   LDX    #PROMPT-1 output message
01160  5008 BD     9BDB               JSR    MESSAG
01170  500B BD     D807               JSR    LINEIN
01180  500E 6D     01                 TST    1,X      if null then end of program
01190  5010 27     4B    505D         BEQ    END
01200  5012 34     10                 PSHS   X
01210  5014 0F     BA                 CLR    $BA      clear format flag
01220  5016 8E     50C2               LDX    #MES0-1
01230  5019 BD     9BDB               JSR    MESSAG
01240  501C BD     D072               JSR    IN
01250  501F BD     D08E               JSR    OUT
01260  5022 81     79                 CMPA   #'y
01270  5024 26     20    5046         BNE    NOTFM    no,not format
01280  5026 8E     50D3               LDX    #MES1-1 set format flag
01290  5029 BD     9BDB               JSR    MESSAG
01300  502C 8D     35    5063         BSR    GETBIN
01310  502E CA     80                 ORB    #$80     format enable
01320  5030 D7     BA                 STB    $BA
01330  5032 8E     5103               LDX    #MES2-1 left of point
01340  5035 BD     9BDB               JSR    MESSAG
01350  5038 8D     54    508E         BSR    GETNUM
01360  503A D7     B9                 STB    $B9
01370  503C 8E     5116               LDX    #MES3-1 rigth of point
01380  503F BD     9BDB               JSR    MESSAG
01390  5042 8D     4A    508E         BSR    GETNUM
01400  5044 D7     B8                 STB    $B8
01410  5046 8E     5129        NOTFM  LDX    #MES4-1
01420  5049 BD     9BDB               JSR    MESSAG
01430  504C 35     10                 PULS   X
01440  504E 9F     D9                 STX    $D9      set pointer
01450  5050 BD     B50B               JSR    STRFAC   string=>FAC1
01460  5053 BD     B622               JSR    FACSTR   FAC1=>string (with format)
01470  5056 30     1F                 LEAX   -1,X
01480  5058 BD     9BDB               JSR    MESSAG   output string
```

```
01490  505B 20   A8   5005          BRA    NEXT
01500                        *
01510  505D BE   513D        END    LDX    TEMP
01520  5060 9F   D9                 STX    $D9
01530  5062 39                      RTS
01540                        *
01550                        * subroutine:get binary number in Breg
01560                        *
01570  5063 5F               GETBIN CLRB
01580  5064 34   04                 PSHS   B
01590  5066 BD   D072        GB01   JSR    IN
01600  5069 81   30                 CMPA   #'0
01610  506B 27   04   5071          BEQ    GB02
01620  506D 81   31                 CMPA   #'1
01630  506F 26   F5   5066          BNE    GB01
01640  5071 BD   D08E        GB02   JSR    OUT
01650  5074 80   30                 SUBA   #'0
01660  5076 46                      RORA
01670  5077 69   E4                 ROL    ,S
01680  5079 5C                      INCB
01690  507A C1   05                 CMPB   #5        skip bit 1
01700  507C 26   0A   5088          BNE    GB03
01710  507E 5C                      INCB
01720  507F 1C   FE                 ANDCC  #$FE
01730  5081 69   E4                 ROL    ,S
01740  5083 86   30                 LDA    #'0
01750  5085 BD   D08E               JSR    OUT
01760  5088 C1   07          GB03   CMPB   #7
01770  508A 26   DA   5066          BNE    GB01
01780  508C 35   84                 PULS   B,PC
01790                        *
01800                        * subroutine:get number in Breg
01810                        *
01820  508E 4F               GETNUM CLRA
01830  508F 5F                      CLRB
01840  5090 34   02          GN01   PSHS   A
01850  5092 86   0A                 LDA    #10       Breg=Breg*10+keyin
01860  5094 3D                      MUL
01870  5095 EB   E0                 ADDB   ,S+
01880  5097 34   04                 PSHS   B
01890  5099 BD   D072               JSR    IN        keyinput
01900  509C 35   04                 PULS   B
01910  509E 81   30                 CMPA   #'0       test '0'..'9'
01920  50A0 25   0B   50AD          BCS    GN02
01930  50A2 81   39                 CMPA   #'9
01940  50A4 22   07   50AD          BHI    GN02
01950  50A6 BD   D08E               JSR    OUT       echo back
01960  50A9 80   30                 SUBA   #'0
01970  50AB 20   E3   5090          BRA    GN01
01980  50AD 39               GN02   RTS
01990                        *
02000  50AE      0D0A        PROMPT FDB    $0D0A,$0D0A
02010  50B2      20                 FCC    '          number ='
02020  50C2      00                 FCB    0
02030  50C3      20          MES0   FCC    ' using ? (y/n) ='
02040  50D3      00                 FCB    0
02050  50D4      0D0A        MES1   FDB    $0D0A
02060  50D6      20                 FCC    '         format       ,*¥+- ^'
02070  50EF      0D0A               FDB    $0D0A
02080  50F1      20                 FCC    '        (0 or 1) = 1'
02090  5103      00                 FCB    0
02100  5104      0D0A        MES2   FDB    $0D0A
02110  5106      6C                 FCC    'left of point  ='
02120  5116      00                 FCB    0
02130  5117      0D0A        MES3   FDB    $0D0A
02140  5119      72                 FCC    'right of point ='
02150  5129      00                 FCB    0
02160  512A      0D0A        MES4   FDB    $0D0A
02170  512C      20                 FCC    '          NUMBER ='
02180  513C      00                 FCB    0
02190                        *
02200  513D      0000        TEMP   FDB    0
02210                        *
02220            5000               END    START
```

```
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=513E
PROGRAM ENTRY ADDR=5000


exec &H5000


        number =12345.6789
 using ? (y/n) =y
       format       ,**¥+- ^
      (0 or 1) = 11111000
left of point  =10
right of point =10
        NUMBER =***+¥12,345.678900000

        number =12345.6789
 using ? (y/n) =y
       format       ,**¥+- ^
      (0 or 1) = 10000001
left of point  =10
right of point =10
        NUMBER = 123456789.000000000D-04

        number =

Ready
```

ＦＡＣとメモリ間の数値の移動

ＦＡＣをメモリに格納する，メモリからＦＡＣに移動する等を行います．

◀ＦＡＣをメモリに格納▶

機　能　ＦＡＣ１の数値をメモリに格納します．

パラメータ　ＦＡＣ１←数値
＄１７←数値の型
Ｘレジスタ←格納するアドレス

アドレス　＄Ｂ３３７

レジスタ　ＣＣ，Ａ，Ｕ

解　説　Ｆ－ＢＡＳＩＣが変数への代入に用いているルーチンです．格納形式は下図の様になります．

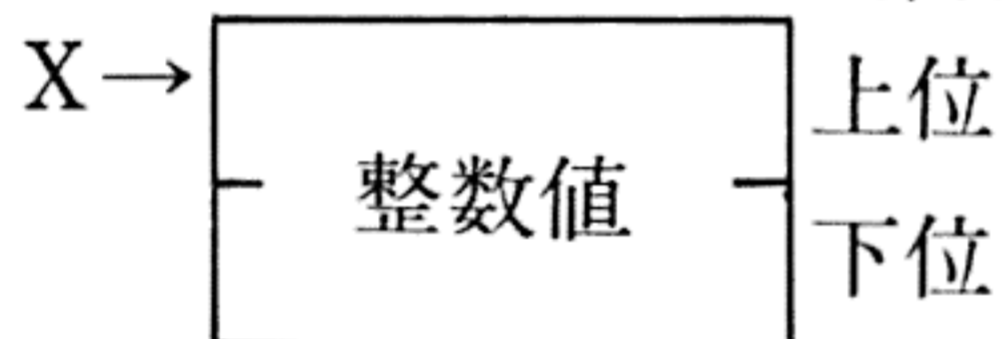
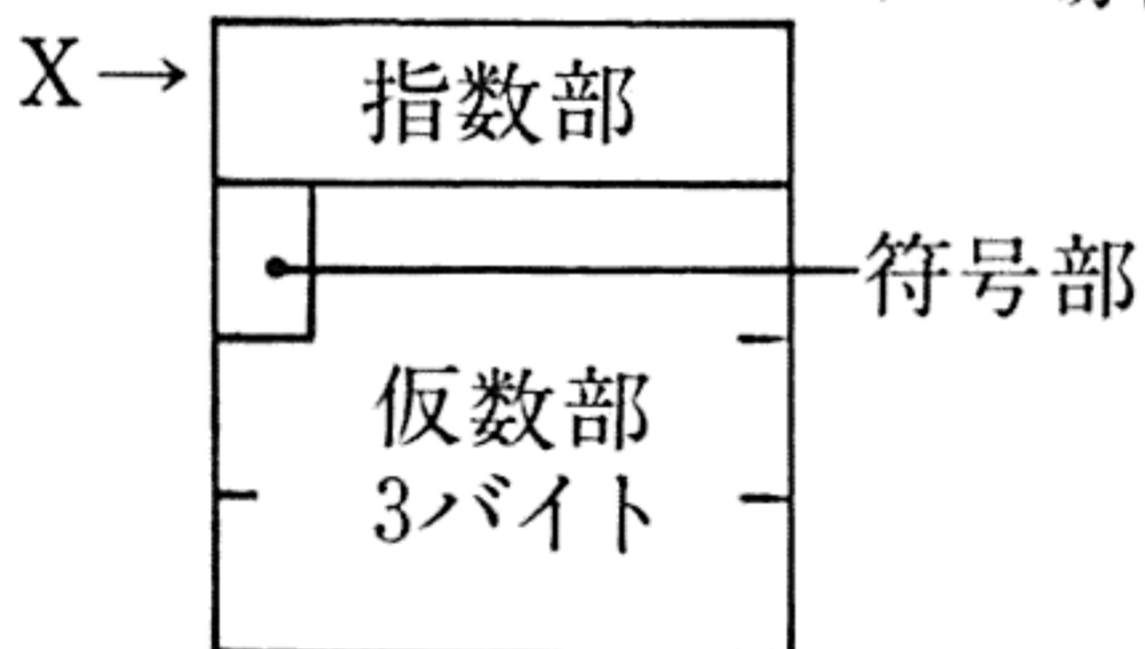
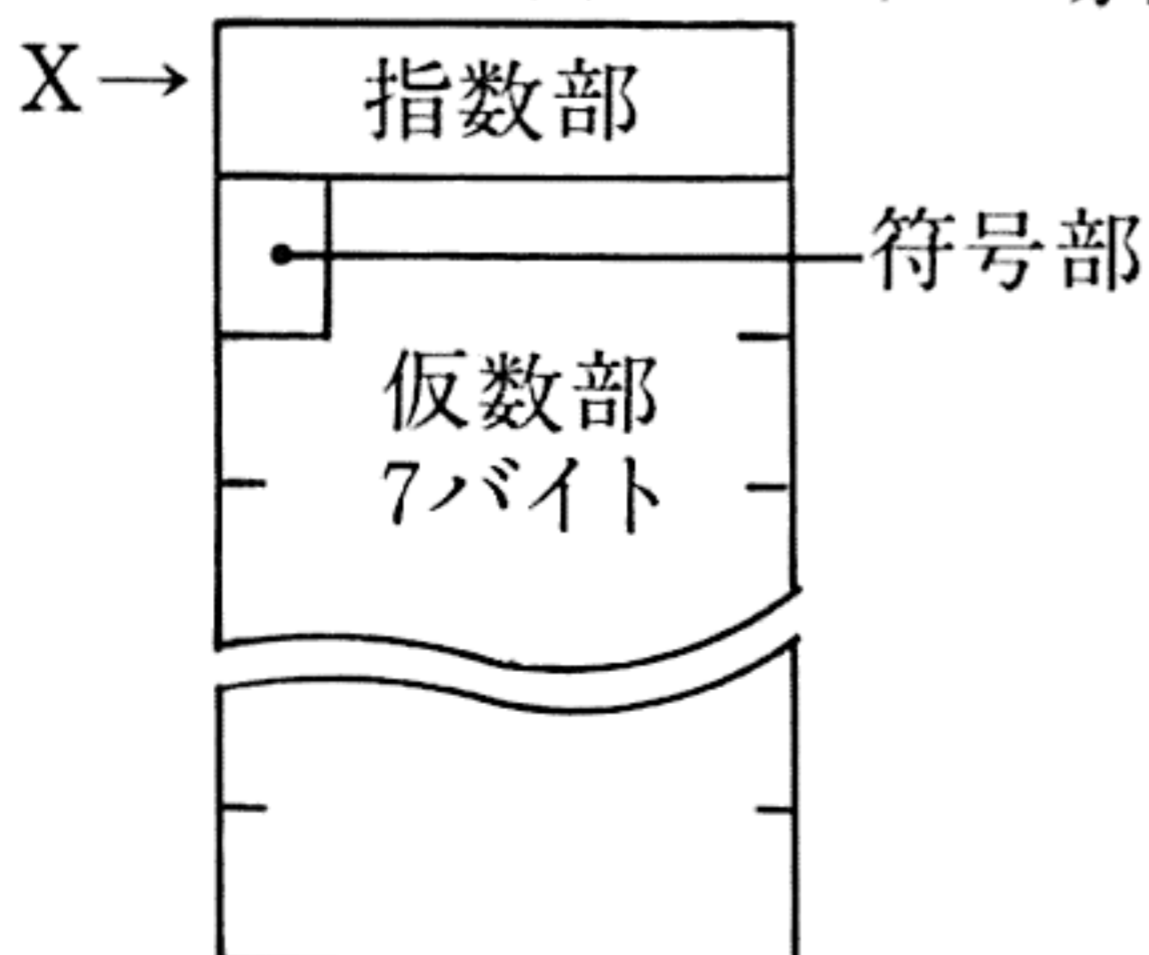
・整数型(＄17＝2)の 場合

X→ 整数値（上位／下位）

・単精度実数(＄17＝4)の 場合

X→ 指数部／符号部／仮数部 3バイト

・倍精度実数(＄17＝8)の 場合

X→ 指数部／符号部／仮数部 7バイト

ＦＡＣ内では，符号部として１バイトを有していますが，メモリ上では仮数部の最上位ビット（ＦＡＣ上では，数値が０を示すとき以外は，１であったビット）を符号ビットとしています．

数値が０を示すときは，すべてのバイトを０として示します．

◀メモリ上からＦＡＣに数値を移動(１)▶

機　能	メモリ上に格納されている数値をＦＡＣに移動します．
パラメータ	Ｘレジスタ←数値の先頭アドレス ＄１７←数値の型
アドレス	＄Ｂ３０１
レジスタ	ＣＣ，Ｂ，Ｕ
復帰情報	ＦＡＣ１：数値

解　説　Ｆ－ＢＡＳＩＣが変数からの数値取り出しに用いているルーチンです．格納形式は，「ＦＡＣをメモリに格納」の場合と同様です．

◀メモリ上からＦＡＣに数値を移動(２)▶

機　能	メモリ上に格納されている実数値をＦＡＣに移動します．
パラメータ	Ｘレジスタ←数値（実数に限る）の先頭アドレス ＄１５←倍精度フラグ（単精度＝０，倍精度＝０以外）
アドレス	＄Ｂ１ＢＡ
レジスタ	ＣＣ，Ａ，Ｂ
復帰情報	ＦＡＣ２：数値

解　説　「メモリ上からＦＡＣに数値を移動(1)」が，整数型でも可能であるのに対して，このルーチンは対象が実数値に限られます．メモリ上での格納形式は「ＦＡＣをメモリに格納」の場合と同様です．

整数をＦＡＣに格納他

◀Ｂレジスタの整数値をＦＡＣに格納▶

機能	Ｂレジスタの値を符号なし整数とみなしてＦＡＣに格納します．
パラメータ	Ｂレジスタ←符号なし整数（０～２５５）
アドレス	＄９７４Ｃ
レジスタ	ＣＣ，Ａ
復帰情報	ＦＡＣ１：整数値 ＄１７：２（整数型）

解説　具体的には，Ａレジスタをクリアして＄９７４Ｄからのルーチンを実行します．

◀Ｄレジスタの整数値をＦＡＣに格納▶

機能	Ｄレジスタの値を符号付き整数とみなしてＦＡＣに格納します．
パラメータ	Ｄレジスタ←符号付き整数（－３２７６８～３２７６７）
アドレス	＄９７４Ｄ
レジスタ	ＣＣ，Ａ
復帰情報	ＦＡＣ１：整数値 ＄１７：２（整数型）

解説　具体的には，Ｄレジスタを＄７６，＄７７にストアし，２を＄１７にストアします．

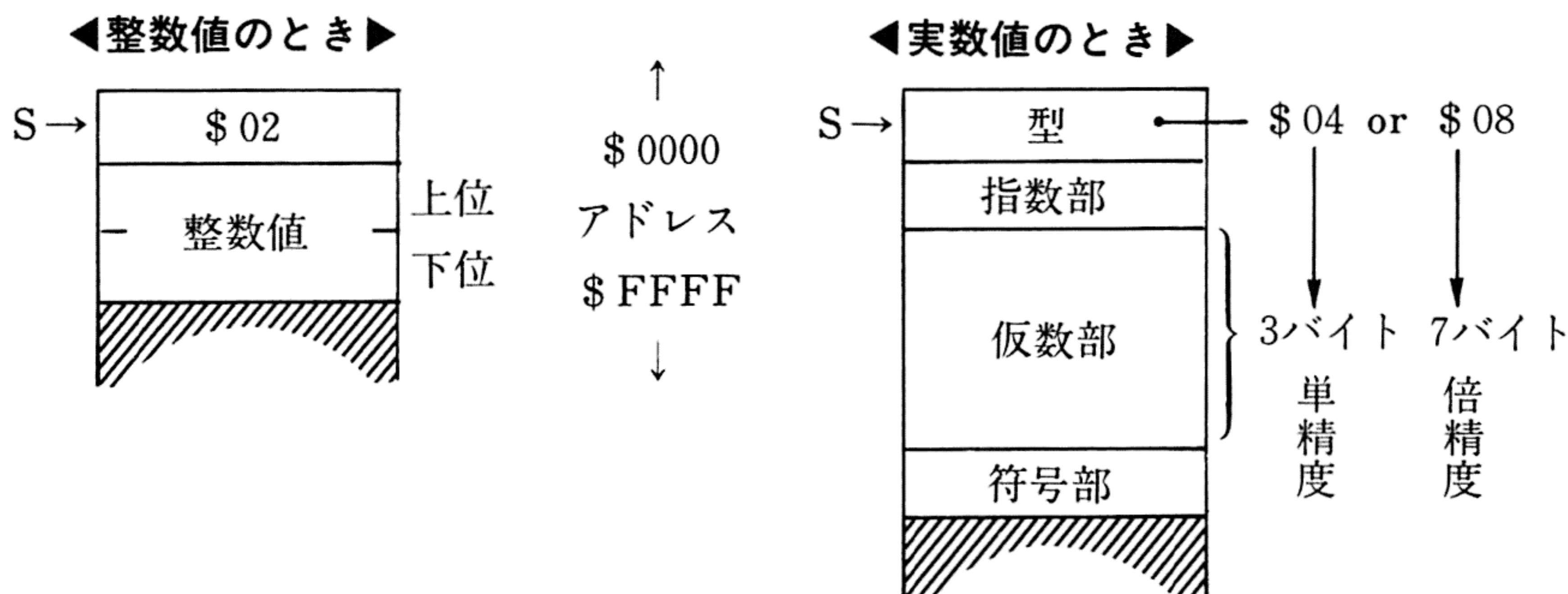
◀Ｄレジスタの整数値を１０進で出力▶

機能	Ｄレジスタの値を符号なし整数とみなして１０進を出力します．
パラメータ	Ｄレジスタ←符号なし整数（０～６５５３６）
アドレス	＄Ｂ６１５
レジスタ	ＣＣ，Ａ，Ｂ，Ｘ，Ｙ，Ｕ

解説　具体的には，順に，＄９７４Ｄ，＄８Ｄ１９，＄Ｂ６２０，＄９ＢＤＢのルーチンを実行します．Ｆ－ＢＡＳＩＣが行番号の出力等に用いているルーチンです．

ＦＡＣとスタック間での数値の移動

◀ＦＡＣ１をスタックにプッシュする▶

機　　能	ＦＡＣ１に格納されている数値をスタックに退避します．
パラメータ	ＦＡＣ１←数値 ＄１７←数値の型
アドレス	＄９３Ｅ８
レジスタ	ＣＣ，Ａ，Ｂ，Ｘ，Ｓ
復帰情報	

解　　説　ＦＡＣ１をスタック上に退避します．この際スタックポインタ(Ｓ)が破壊されますが，このルーチンからの戻りは正常に行われます．

◀スタックからＦＡＣ２へポップする▶

機　　能	ＦＡＣ２にスタックから数値を復帰します．
パラメータ	スタックの状態は，プッシュした後と同様でなければなりません．
アドレス	＄９４１１
レジスタ	ＣＣ，Ａ，Ｂ，Ｘ，Ｓ
復帰情報	ＦＡＣ２：数値 ＄５Ｄ：数値の型

解　　説　前のルーチンで退避した数値をＦＡＣ２に復帰します．この際スタックポインタ（Ｓ）が破壊されますが，このルーチンからの戻りは正常に行われます．これらのルーチンは，システムスタックを使用するので，サブルーチンからの戻り等には注意して下さい．ＪＳＲ＄９３Ｅ８を「ＰＳＨＳ　ＦＡＣ１」，ＪＳＲ＄９４１１を「ＰＵＬＳ　ＦＡＣ２」と考えると，容易なものとなります．

ＦＡＣ同志での数値の移動

◀ＦＡＣ１⇒ＦＡＣ２▶

機　能	ＦＡＣ１に格納されている数値をＦＡＣ２に代入します．
パラメータ	ＦＡＣ１←数値 ＄１７←数値の型
アドレス	＄Ｂ３８０
レジスタ	ＣＣ，Ａ，Ｘ
復帰情報	ＦＡＣ２：ＦＡＣ１と同じ数値

解　説　ＦＡＣ１の数値をＦＡＣ２に代入します．ＦＡＣ２の型は，ＦＡＣ１のそれと同様になりますが，ＦＡＣ２の型を示すアドレス（＄５Ｄ）に型はセットされません．

◀ＦＡＣ２⇒ＦＡＣ１▶

機　能	ＦＡＣ２に格納されている数値をＦＡＣ２に代入します．
パラメータ	ＦＡＣ２←数値 ＄１７←数値の型
アドレス	＄Ｂ３５Ｆ
レジスタ	ＣＣ，Ａ，Ｘ
復帰情報	ＦＡＣ１：ＦＡＣ２と同じ数値

解　説　ＦＡＣ２の数値をＦＡＣ１に代入します．ＦＡＣ２の型を示すアドレス（＄５Ｄ）は参照されず，ＦＡＣ１のそれ（＄１７）が参照されます．

◀ＦＡＣ１⇔ＦＡＣ２▶

機　能	ＦＡＣ１の数値とＦＡＣ２の数値を入れ換えます．
パラメータ	ＦＡＣ１←数値１ ＦＡＣ２←数値２
アドレス	数値が倍精度実数のとき　＄９３６４ 数値が単精度実数のとき　＄９３７４ 数値が整数のとき　＄９３８４
レジスタ	ＣＣ，Ａ，Ｂ，Ｘ

復帰情報　ＦＡＣ１：数値２
ＦＡＣ２：数値１

解　　説　ＦＡＣ１の数値とＦＡＣ２の数値を交換します．ただし，ＦＡＣ１の数値の型と，ＦＡＣ２のそれは，一致していなければなりません．

ＦＡＣの数値の判別

◀ＦＡＣ１の数値の型を判別▶

機　能　ＦＡＣ１に格納されている数値の型を判別します．

パラメータ　＄１７←数値の型

アドレス　＄ＢＣＢ３

レジスタ　ＣＣ，Ａ

復帰情報　型の情報はＣＣレジスタの各フラグに現われます．
倍精度実数（８）のときキャリー（Ｃ：bit０）リセット
単精度実数（４）のときオーバーフロー（Ｖ：bit１）セット
整数（２）のときネガティブ（Ｎ：bit３）セット
文字列（３）のときゼロ（Ｚ：bit２）セット

解　説　＄１７の値を参照して各フラグをセットします．

◀ＦＡＣ１の符号を判別▶

機　能　ＦＡＣ１に格納されている数値の符号を判別します．

パラメータ　ＦＡＣ１←数値
＄１７←数値の型

アドレス　＄Ｂ３Ｃ８

レジスタ　ＣＣ，Ａ，Ｂ，Ｘ

復帰情報　Ｂレジスタ：数値が正のとき　＄０１
数値が負のとき　＄ＦＦ
数値が０のとき　＄００

解　説　ＦＡＣ１の符号により，Ｂレジスタに値をセットします．

サンプル

```
PAGE  001 (        ,        )

01000                       *
01010                       * output type & sign
01020                       *
01030                              OPT    M,NOS,NOG,P=255
01040              BCB3     TTEST  EQU    $BCB3    type test
01050              B3C8     STEST  EQU    $B3C8    sign test
01060              D08E     OUT    EQU    $D08E    output 1 character
01070  5000                        ORG    $5000
01080              5000     START  EQU    *
01090                       *
01100  5000 97     17              STA    $17      set type flag
01110  5002 BD     BCB3            JSR    TTEST    type test
01120  5005 24     0A    5011      BCC    DBL
01130  5007 29     0E    5017      BVS    SNG
```

```
01140  5009 27   2D   5038         BEQ    STR
01150  500B 30   8D 0073           LEAX   MESINT,PCR
01160  500F 20   0A   501B         BRA    TO1
01170  5011 30   8D 0050   DBL     LEAX   MESDBL,PCR
01180  5015 20   04   501B         BRA    TO1
01190  5017 30   8D 002D   SNG     LEAX   MESSNG,PCR
01200  501B 8D   21   503E TO1     BSR    MESSAG
01210                      *
01220  501D BD   B3C8              JSR    STEST    sign test
01230  5020 5D                     TSTB
01240  5021 27   08   502B         BEQ    ZERO
01250  5023 2B   0C   5031         BMI    MINUS
01260  5025 30   8D 0077           LEAX   MESPLS,PCR
01270  5029 20   0A   5035         BRA    SO1
01280  502B 30   8D 007C   ZERO    LEAX   MESZER,PCR
01290  502F 20   04   5035         BRA    SO1
01300  5031 30   8D 007D   MINUS   LEAX   MESMIN,PCR
01310  5035 8D   07   503E SO1     BSR    MESSAG
01320  5037 39                     RTS
01330                      *
01340  5038 30   8D 005A   STR     LEAX   MESSTR,PCR
01350  503C 20   F7   5035         BRA    SO1
01360                      *
01370                      * subroutine:output 1 line
01380                      *
01390  503E A6   80        MESSAG LDA     ,X+
01400  5040 27   05   5047         BEQ    MO1
01410  5042 BD   D0BE              JSR    OUT
01420  5045 20   F7   503E         BRA    MESSAG
01430  5047 39             MO1     RTS
01440                      *
01450  5048      0D0A      MESSNG FDB     $0D0A
01460  504A      73                FCC    'single precision number of'
01470  5064      00                FCB    0
01480  5065      0D0A      MESDBL FDB     $0D0A
01490  5067      64                FCC    'double precision number of'
01500  5081      00                FCB    0
01510  5082      0D0A      MESINT FDB     $0D0A
01520  5084      69                FCC    'integer number of'
01530  5095      00                FCB    0
01540  5096      0D0A      MESSTR FDB     $0D0A
01550  5098      73                FCC    'string.'
01560  509F      00                FCB    0
01570                      *
01580  50A0      20        MESPLS FCC     ' positive.'
01590  50AA      00                FCB    0
01600  50AB      20        MESZER FCC     ' zero.'
01610  50B1      00                FCB    0
01620  50B2      20        MESMIN FCC     ' negative.'
01630  50BC      00                FCB    0
01640                      *
01650            5000              END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50BC
PROGRAM ENTRY ADDR=5000

def usr=&H5000

Ready
?usr(10%):?usr(0!):?usr(-10#)

integer number of positive. 10

single precision number of zero. 0

double precision number of negative.-10

Ready
?usr("*")
string.*
Ready
```

ＦＡＣの型を変換

◀任意の型に変換▶

機　　能	ＦＡＣ１の数値を指定した型の数値へと変換します．
パラメータ	ＦＡＣ１←数値 ＄１７←数値の型（旧） Ａレジスタ←数値の型（新）
アドレス	＄ＢＣＣ１
レジスタ	ＣＣ，Ａ，Ｂ，Ｘ，Ｕ
復帰情報	ＦＡＣ１：変換された数値 ＄１７：数値の型（新）

解　　説　Ａレジスタで指定した型へ，ＦＡＣ１の数値を変換します．

◀型の相互変換▶

機　　能	ＦＡＣ１の型を変換します．
パラメータ	ＦＡＣ１←数値 ＄１７←数値の型（旧）（下表の＊＊のルーチンの時だけ必要）
アドレス	

旧＼新	整数型	単精度実数型	倍精度実数型
整数型		＄BCFF＊	＄BCD3
単精度実数型			＄BCD5
倍精度実数型		＄BCEE	
＊＊	＄BC74	＄BCE０	＄BCCD

＊：符号付きの場合、符号なしの場合は＄BD19
＊＊：＄17にセットされている型を参照して処理を行う。これ以外の所では、＄17に値をセットする必要はありません．

レジスタ	ＣＣ，Ａ，Ｂ，Ｘ，Ｕ
復帰情報	ＦＡＣ１：変換された数値 ＄１７：数値の型（新）

解　　説　各ルーチンに応じて，ＦＡＣ１の型を変換します．

サンプル

```
PAGE  001 (        ,         )

01000                        *
01010                        * variable type convert
01020                        *
01030                                OPT    M,NOS,NOG,P=255
01040             D807       LINEIN EQU    $D807    line input
01050             D08E       OUT    EQU    $D08E    output 1 character
01060             D072       IN     EQU    $D072    input 1 character
01070             B50B       STRFAC EQU    $B50B    convert string into FAC1
01080             B622       FACSTR EQU    $B622    convert FAC1 into string
01090             BCC1       CONV   EQU    $BCC1    type convert
01100  5000                         ORG    $5000
01110             5000       START  EQU    *
01120                        *
01130  5000 9E    D9                LDX    $D9      save POINTER
01140  5002 BF    5080              STX    TEMP
01150  5005 8E    504E       NEXT   LDX    #PROMPT output message
01160  5008 8D    3A    5044        BSR    MESSAG
01170  500A BD    D807              JSR    LINEIN   get number
01180  500D 6D    01                TST    1,X      if null then end of program
01190  500F 27    2D    503E        BEQ    END
01200  5011 9F    D9                STX    $D9
01210  5013 BD    B50B              JSR    STRFAC   string=>FAC1
01220  5016 8E    505F              LDX    #MES
01230  5019 8D    29    5044        BSR    MESSAG
01240  501B BD    D072       NO1    JSR    IN       get type of var.
01250  501E 81    38                CMPA   #'8
01260  5020 27    08    502A        BEQ    NO2
01270  5022 81    34                CMPA   #'4
01280  5024 27    04    502A        BEQ    NO2
01290  5026 81    32                CMPA   #'2
01300  5028 26    F1    501B        BNE    NO1
01310  502A BD    D08E       NO2    JSR    OUT      echo back
01320  502D 80    30                SUBA   #'0
01330  502F BD    BCC1              JSR    CONV     type convert
01340  5032 8E    506F              LDX    #ANSWER
01350  5035 8D    0D    5044        BSR    MESSAG
01360  5037 BD    B622              JSR    FACSTR   output FAC1
01370  503A 8D    08    5044        BSR    MESSAG
01380  503C 20    C7    5005        BRA    NEXT
01390                        *
01400  503E BE    5080       END    LDX    TEMP     end
01410  5041 9F    D9                STX    $D9
01420  5043 39                      RTS
01430                        *
01440                        * subroutine:output 1 line
01450                        *
01460  5044 A6    80         MESSAG LDA    ,X+
01470  5046 27    05    504D        BEQ    M01
01480  5048 BD    D08E              JSR    OUT
01490  504B 20    F7    5044        BRA    MESSAG
01500  504D 39               M01    RTS
01510                        *
01520  504E       0D0A       PROMPT FDB    $0D0A
01530  5050       20                FCC    '        number :'
01540  505E       00                FCB    0
01550  505F       74         MES    FCC    'type (2/4/8) ? '
01560  506E       00                FCB    0
01570  506F       0D0A       ANSWER FDB    $0D0A
01580  5071       20                FCC    '        NUMBER :'
01590  507F       00                FCB    0
01600                        *
01610  5080       0000       TEMP   FDB    0
01620                        *
01630             5000              END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5081
PROGRAM ENTRY ADDR=5000
```

```
exec &H5000

       number :123.456
type (2/4/8) ? 2
       NUMBER :  123
       number :123.456
type (2/4/8) ? 8
       NUMBER :  123.4559936523438
       number :1.234567890123456789d-10
type (2/4/8) ? 4
       NUMBER :  1.23457E-10
       number :123456
type (2/4/8) ? 2

Overflow
Ready
```

◀ＦＡＣ１とＦＡＣ２の型をあわせる▶

機　　能	**２項演算に先だって，ＦＡＣ１とＦＡＣ２の型を一致させます．**
パラメータ	ＦＡＣ１←数値(１)　＄１７←数値(１)の型　＄１５←０
	ＦＡＣ２←数値(２)　＄５Ｄ←数値(２)の型
アドレス	＄９３４Ｄ
レジスタ	ＣＣ，Ａ，Ｂ，Ｘ，Ｕ
復帰情報	＄１７←新しい数値の型
	＄１５←倍精度フラグ（０：単精度，０以外：倍精度）

解　　説　ＦＡＣ１とＦＡＣ２の型を比較して，低い精度のものを高い精度のものに変換します．

ＦＡＣ１もＦＡＣ２も整数型であった場合に限り，ＦＡＣ１とＦＡＣ２が交換されます．これは後述する演算ルーチンのための処置です．

2項演算

◀四則演算▶

ＦＡＣ１とＦＡＣ２の間で四則演算を行い，結果をＦＡＣ１に格納します．

〔パターン１〕

パラメータ	ＦＡＣ１←演算数	＄１７←演算数の型（２，４，８）
	ＦＡＣ２←被演算数	＄５Ｄ←被演算数の型（２，４，８）

アドレス
加算……＄ＢＤ４Ｄ
減算……＄ＢＤ３Ｆ
乗算……＄ＢＤ７３
除算……＄９３ＣＤ
（このアドレスを呼び出す前に ＪＳＲ＄９３４Ｄ を実行すること）

レジスタ　ＣＣ，Ａ，Ｂ，Ｘ，Ｕ

復帰情報　ＦＡＣ１：結果
＄１７：結果の型

解　説　ＦＡＣ１⇐ＦＡＣ２◎ＦＡＣ１　を実行します．数値の型は限定されません．

〔パターン２〕

パラメータ　ＦＡＣ１←演算数（実数）
ＦＡＣ２←被演算数（実数）
＄１５←倍精度フラグ（０：単精度，０以外：倍精度）
Ａレジスタ←被演算数の指数部（＄８２）
Ｂレジスタ←演算数の指数部（＄７４）
＄８Ｂ←ＦＡＣ１とＦＡＣ２の符号部のＥＯＲ

アドレス　必ず　ＬＤＢ　＄７４
ＪＳＲ　＄ＸＸＸＸ　の順で実行すること．
加算……＄ＡＦ１Ｄ
減算……＄ＡＦ１４
乗算……＄Ｂ０Ｆ１
除算……＄Ｂ２３６

レジスタ　ＣＣ，Ａ，Ｂ，Ｘ，Ｕ

復帰情報　ＦＡＣ１：結果

解　説　FAC1⇦FAC2◎FAC1　を実行します．数値は同じ精度の実数に限られます．

〔パターン3〕

パラメータ　FAC1←演算数（実数）
Xレジスタ←被演算数（実数）の先頭アドレス
＄15←倍精度フラグ（0：単精度，0以外：倍精度）

アドレス　加算――＄AF1A
減算――＄AF11
乗算――＄B0EE
除算――＄B234

レジスタ　CC，A，B，X，U

復帰情報　FAC1：結果

解　説　FAC1⇦（X）◎FAC1　を実行します．数値は，同じ精度の実数に限られます．被演算数（実数）は，「FACをメモリに格納」と同じ形でメモリ上にあるデータです．

〔パターン4〕

パラメータ　FAC1←被演算数（整数）　＄17←2（整数型）
FAC2←演算数（整数）

アドレス　加算――＄BD52
減算――＄BD44
乗算――＄BD78
除算――＄93D3

レジスタ　CC，A，B，X，U

復帰情報　FAC1：結果　＄17：結果の型

解　説　FAC1⇦FAC1◎FAC2　を実行します．この場合だけ，演算数と被演算数の順序が逆ですので注意して下さい．結果が整数型で収まらない場合と除算の場合は，結果が単精度実数で返されます．

サンプル　スタック計算機をシミュレートするプログラムです．数値または，演算子（＋，－，＊，／）をカンマまたはCRでくぎって入力して下さい。'＝'はスタックトップの値を表示します．

```
PAGE   001  (          ,          )

01000                             *
01010                             * stack calculator
01020                             *
01030                                    OPT     M,NOS,NOG,P=255
01040               D807          LINEIN EQU     $D807    line input
01050               9BDB          MESSAG EQU     $9BDB    output from (X+1) till 0
01060               9B50          CRLF   EQU     $9B50    output CR & LF
01070               934D          FACSET EQU     $934D    type set (FAC1=FAC2)
01080               93CD          DIV    EQU     $93CD    FAC2/FAC1=>FAC1
01090               93E8          FACPSH EQU     $93E8    push FAC1
01100               9411          FACPUL EQU     $9411    pull FAC2
01110               B35F          FAC2.1 EQU     $B35F    FAC2=>FAC1
01120               B506          STRFAC EQU     $B506    convert string into FAC1
01130               B620          FACSTR EQU     $B620    convert FAC1 into string
01140               BD3F          SUB    EQU     $BD3F    FAC2-FAC1=>FAC1
01150               BD4D          ADD    EQU     $BD4D    FAC2+FAC1=>FAC1
01160               BD73          MUL    EQU     $BD73    FAC2*FAC1=>FAC1
01170  5000                              ORG     $5000
01180               5000          START  EQU     *
01190                             *
01200  5000 9E      D9                   LDX     $D9      save POINTER
01210  5002 BF      5117                 STX     TEMP
01220  5005 BD      9B50                 JSR     CRLF
01230  5008 8E      50E8          NEXT   LDX     #PROMPT-1 output message
01240  500B BD      9BDB                 JSR     MESSAG
01250  500E BD      D807                 JSR     LINEIN
01260  5011 6D      01                   TST     1,X      if null then end of program
01270  5013 1027    00B1 50C8            LBEQ    END
01280  5017 A6      01            GET    LDA     1,X      get next character
01290  5019 27      ED   5008            BEQ     NEXT
01300  501B 81      2D                   CMPA    #'-      '-' ?
01310  501D 26      0C   502B            BNE     G01
01320  501F 6D      02                   TST     2,X      yes
01330  5021 27      08   502B            BEQ     G01
01340  5023 A6      02                   LDA     2,X
01350  5025 81      2C                   CMPA    #',
01360  5027 26      61   508A            BNE     G05      if monadic operator
01370  5029 86      2D                   LDA     #'-
01380  502B 81      2C            G01    CMPA    #',      delimiter ?
01390  502D 26      04   5033            BNE     G02
01400  502F 30      01                   LEAX    1,X      yes,skip ','
01410  5031 20      E4   5017            BRA     GET
01420  5033 CE      510B          G02    LDU     #TABLE   operator check
01430  5036 A1      C0            G03    CMPA    ,U+
01440  5038 26      30   506A            BNE     G04
01450  503A 30      01                   LEAX    1,X      found ! skip 1 chr
01460  503C 9F      D9                   STX     $D9      save pointer
01470  503E AE      C4                   LDX     ,U
01480  5040 BF      511A                 STX     ADDR     set address of operator
01490  5043 B6      5119                 LDA     STACK    stack test
01500  5046 81      02                   CMPA    #2
01510  5048 1025    0087 50D3            LBLO    ERR
01520  504C BD      9411                 JSR     FACPUL   stack=>FAC2
01530  504F 96      5D                   LDA     $5D      FAC2=>FAC1
01540  5051 97      17                   STA     $17
01550  5053 BD      B35F                 JSR     FAC2.1
01560  5056 BD      9411                 JSR     FACPUL   stack=>FAC2
01570  5059 BD      934D                 JSR     FACSET   type set
01580  505C AD      9F 511A              JSR     [ADDR]   execute operator
01590  5060 BD      93E8                 JSR     FACPSH   FAC1(ans)=>stack
01600  5063 7A      5119                 DEC     STACK    stack:-1
01610  5066 9E      D9                   LDX     $D9      pointer come back
01620  5068 20      AD   5017            BRA     GET
01630  506A 33      42            G04    LEAU    2,U
01640  506C 1183    5117                 CMPU    #.TABLE
01650  5070 26      C4   5036            BNE     G03
01660                             *
01670  5072 81      3D                   CMPA    #'=      display '=' ?
01680  5074 27      26   509C            BEQ     DSP
01690  5076 81      26                   CMPA    #'&      number test
01700  5078 27      10   508A            BEQ     G05
01710  507A 81      2E                   CMPA    #'.
```

```
01720  507C 27   0C   508A         BEQ    G05
01730  507E 81   2D                CMPA   #'-
01740  5080 27   08   508A         BEQ    G05
01750  5082 81   30                CMPA   #'0
01760  5084 25   4D   50D3         BLO    ERR      if not number then error
01770  5086 81   39                CMPA   #'9
01780  5088 22   49   50D3         BHI    ERR
01790  508A 9F   D9        G05     STX    $D9      number
01800  508C BD   B506              JSR    STRFAC   string=>FAC1
01810  508F BD   93E8              JSR    FACPSH   FAC1=>stack
01820  5092 7C   5119              INC    STACK    stack:+1
01830  5095 9E   D9                LDX    $D9
01840  5097 30   1F                LEAX   -1,X
01850  5099 16   FF7B 5017         LBRA   GET
01860                      *
01870  509C 9F   D9        DSP     STX    $D9
01880  509E 7D   5119              TST    STACK    test stack
01890  50A1 27   30   50D3         BEQ    ERR
01900  50A3 8E   50ED              LDX    #MES-1   output message
01910  50A6 BD   9BDB              JSR    MESSAG
01920  50A9 BD   9411              JSR    FACPUL   stack=>FAC2
01930  50AC 96   5D                LDA    $5D      FAC2=>FAC1
01940  50AE 97   17                STA    $17
01950  50B0 BD   B35F              JSR    FAC2.1
01960  50B3 BD   B620              JSR    FACSTR   FAC1=>string
01970  50B6 30   1F                LEAX   -1,X     output string
01980  50B8 BD   9BDB              JSR    MESSAG
01990  50BB BD   9B50              JSR    CRLF
02000  50BE 7A   5119              DEC    STACK    stack:-1
02010  50C1 9E   D9                LDX    $D9
02020  50C3 30   01                LEAX   1,X
02030  50C5 16   FF4F 5017         LBRA   GET
02040                      *
02050  50C8 7D   5119      END     TST    STACK    end of exec
02060  50CB 26   06   50D3         BNE    ERR      stack check
02070  50CD BE   5117      END01   LDX    TEMP     POINTER come back
02080  50D0 9F   D9                STX    $D9
02090  50D2 39                     RTS
02100                      *
02110  50D3 8E   50FA      ERR     LDX    #ERRMES-1 error
02120  50D6 BD   9BDB              JSR    MESSAG   output messag
02130  50D9 7D   5119              TST    STACK    stack test
02140  50DC 27   08   50E6         BEQ    ERR02
02150  50DE BD   9411      ERR01   JSR    FACPUL   stack recover
02160  50E1 7A   5119              DEC    STACK
02170  50E4 26   FB   50DE         BNE    ERR01
02180  50E6 16   FF1F 5008 ERR02   LBRA   NEXT     get next line
02190                      *
02200  50E9      3F        PROMPT  FCC    '??? '
02210  50ED      00                FCB    0
02220  50EE      20        MES     FCC    '     Answer:'
02230  50FA      00                FCB    0
02240  50FB      2A        ERRMES  FCC    '*** ERROR ***'
02250  5108      0D0A              FDB    $0D0A
02260  510A      00                FCB    0
02265                      *
02270  510B      2B        TABLE   FCC    '+'      operator table
02280  510C      BD4D              FDB    ADD
02290  510E      2D                FCC    '-'
02300  510F      BD3F              FDB    SUB
02310  5111      2A                FCC    '*'
02320  5112      BD73              FDB    MUL
02330  5114      2F                FCC    '/'
02340  5115      93CD              FDB    DIV
02350            5117      .TABLE  EQU    *
02360                      *
02370  5117      0000      TEMP    FDB    0
02380  5119      00        STACK   FCB    0
02390  511A      0000      ADDR    FDB    0
02400                      *
02410            5000              END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```

```
PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=511B
PROGRAM ENTRY ADDR=5000


exec &H5000

??? 10,20,30,+,*,=
     Answer: 500
??? 40,50,30,20
??? /,=,-,
     Answer: 1.5
??? =
     Answer:-10
??? &H1000,&H1000,+,=
     Answer: 8192
??? 1.23456789012345
??? 1000
??? /,=
     Answer: 1.23456789012345D-03
??? 20,6,/,=
     Answer: 3.333333333333333
???

Ready
HARDC
```

◀整数除算及び剰余▶

機　能　整数除算及び剰余を求めます．

パラメータ　ＦＡＣ１←被演算数（整数）
　　ＦＡＣ２←演算数（整数）

アドレス　整数除算——＄ＢＥ０Ｃ
　　剰余——＄ＢＥ３３（ただしＡレジスタに＄７６の値をセットしてから実行すること）

レジスタ　ＣＣ，Ａ，Ｂ，Ｘ

復帰情報　ＦＡＣ１：結果（整数）

解　説　ＦＡＣ１⇐ＦＡＣ１◎ＦＡＣ２　を実行します．数値は整数に限られます．

サンプル

```
PAGE   001  (          ,          )

01000                          *
01010                          * div(¥) & mod
01020                          *
01030                                  OPT     M,NOG,NOS,P=255
01040                D072      IN      EQU     $D072
01050                D08E      OUT     EQU     $D08E
01060                BE0C      INTDIV  EQU     $BE0C
01070                BE33      MOD     EQU     $BE33
01080                B622      FACSTR  EQU     $B622
01090    5000                          ORG     $5000
01100                5000      START   EQU     *
01110                          *
01120    5000 8E     5092              LDX     #MES1
01130    5003 8D     3D    5042        BSR     MESSAG
01140    5005 8D     45    504C        BSR     GETNUM
01150    5007 DD     76                STD     $76      set FAC1
01160    5009 34     06                PSHS    D
01170    500B 8E     509D              LDX     #MES2
01180    500E 8D     32    5042        BSR     MESSAG
01190    5010 8D     3A    504C        BSR     GETNUM
01200    5012 DD     84                STD     $84      set FAC2
01210    5014 34     06                PSHS    D
01220    5016 86     02                LDA     #2       set INT
01230    5018 97     17                STA     $17
01240                          *
01250    501A BD     BE0C              JSR     INTDIV   FAC1<=FAC1 ¥ FAC2
01260    501D 8E     50A8              LDX     #MES3
01270    5020 8D     20    5042        BSR     MESSAG
01280    5022 BD     B622              JSR     FACSTR   output FAC1
01290    5025 8D     1B    5042        BSR     MESSAG
01300                          *
01310    5027 86     02                LDA     #2       set INT
01320    5029 97     17                STA     $17
01330    502B 35     06                PULS    D
01340    502D DD     84                STD     $84      set FAC2
01350    502F 35     06                PULS    D
01360    5031 DD     76                STD     $76      set FAC1
01370    5033 BD     BE33              JSR     MOD      FAC1<=FAC1 MOD FAC2
01380    5036 8E     50B3              LDX     #MES4
01390    5039 8D     07    5042        BSR     MESSAG
01400    503B BD     B622              JSR     FACSTR   output FAC1
01410    503E 8D     02    5042        BSR     MESSAG
01420    5040 20     BE    5000        BRA     START
```

```
01430                               *
01440                               * output 1 line
01450                               *
01460  5042 A6    80                MESSAG LDA     ,X+
01470  5044 27    05     504B              BEQ     ME01
01480  5046 BD    D08E                     JSR     OUT
01490  5049 20    F7     5042              BRA     MESSAG
01500  504B 39                      ME01   RTS
01510                               *
01520                               * get number in Dreg
01530                               *
01540  504C CC    0000              GETNUM LDD     #0
01550  504F 7F    50B7                     CLR     FLAG
01560  5052 34    06                       PSHS    D
01570  5054 4F                             CLRA
01580  5055 1F    89                GN01   TFR     A,B
01590  5057 4F                             CLRA
01600  5058 34    06                       PSHS    D
01610  505A EC    62                       LDD     2,S
01620  505C 58                             ASLB
01630  505D 49                             ROLA
01640  505E 58                             ASLB
01650  505F 49                             ROLA
01660  5060 E3    62                       ADDD    2,S
01670  5062 58                             ASLB
01680  5063 49                             ROLA
01690  5064 E3    E1                       ADDD    ,S++
01700  5066 ED    E4                       STD     ,S
01710  5068 BD    D072              GN02   JSR     IN
01720  506B 81    2D                       CMPA    #'-
01730  506D 27    1B     508A              BEQ     GN05
01740  506F 81    30                       CMPA    #'0
01750  5071 25    0B     507E              BLO     GN03
01760  5073 81    39                       CMPA    #'9
01770  5075 22    07     507E              BHI     GN03
01780  5077 BD    D08E                     JSR     OUT
01790  507A 80    30                       SUBA    #'0
01800  507C 20    D7     5055              BRA     GN01
01810  507E 35    06                GN03   PULS    D
01820  5080 7D    50B7                     TST     FLAG
01830  5083 27    04     5089              BEQ     GN04
01840  5085 40                             NEGA
01850  5086 50                             NEGB
01860  5087 82    00                       SBCA    #0
01870  5089 39                      GN04   RTS
01880  508A 7C    50B7              GN05   INC     FLAG
01890  508D BD    D08E                     JSR     OUT
01900  5090 20    D6     5068              BRA     GN02
01910                               *
01920  5092       0D0A              MES1   FDB     $0D0A
01930  5094       20                       FCC     '         X%='
01940  509C       00                       FCB     0
01950  509D       0D0A              MES2   FDB     $0D0A
01960  509F       20                       FCC     '         Y%='
01970  50A7       00                       FCB     0
01980  50A8       0D0A              MES3   FDB     $0D0A
01990  50AA       58                       FCC     'X% / Y%='
02000  50B2       00                       FCB     0
02010  50B3       2E                MES4   FCC     '...'
02020  50B6       00                       FCB     0
02030                               *
02040  50B7       00                FLAG   FCB     0
02050                               *
02060             5000                     END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50B7
PROGRAM ENTRY ADDR=5000

exec &H5000
```

```
         X%=12
         Y%=5
X% / Y%= 2... 2
         X%=-30
         Y%=4
X% / Y%=-7...-2
         X%=0
         Y%=5
X% / Y%= 0... 0
         X%=0
         Y%=0

Division By Zero
Ready
```

◀FACの比較▶

機　能　FAC1とFAC2の値を比較します.

〔パターン1〕

パラメータ　FAC1←数値(1)　$17←数値(1)の型（2, 4, 8）
FAC2←数値(2)　$5D←数値(2)の型（2, 4, 8）
JSR$934D
JSR$BE44　の順で実行する.

アドレス　CC, A, B, X, U

レジスタ　FAC1−FAC2が負のとき　Bレジスタ：$FF
0のとき　Bレジスタ：$00
正のとき　Bレジスタ：$01

〔パターン2〕

パラメータ　FAC1←数値(1)（実数）
FAC2←数値(2)（実数）
$15←倍精度フラグ（0：単精度, 0以外：倍精度）

アドレス　$B3E1

レジスタ　CC, A, B, X

復帰情報　〔パターン1〕と同様

〔パターン3〕

パラメータ　FAC1←被演算子（整数）
FAC2←演算子（整数）

アドレス　$BE4A

レジスタ　CC, A, B

復帰情報　FAC2−FAC1の正負によりBレジスタをセットします.
Bレジスタ：　負のとき　$FF
0のとき　$00
正のとき　$01

サンプル

```
PAGE  001 (      ,       )

01000                          *
01010                          * compare FAC
01020                          *
01030                                  OPT    M,NOS,NOG,P=255
01040            D807          LINEIN  EQU    $D807   line input
01050            D08E          OUT     EQU    $D08E   output 1 character
01060            D072          IN      EQU    $D072   input 1 character
01070            B50B          STRFAC  EQU    $B50B   convert string into FAC1
01080            B622          FACSTR  EQU    $B622   convert FAC1 into string
01090            934D          FACSET  EQU    $934D   set type of FAC
01100            BE44          FACCMP  EQU    $BE44   FAC compare
01110            93E8          FACPSH  EQU    $93E8   push FAC1
01120            9411          FACPUL  EQU    $9411   pull FAC2
01130  5000                            ORG    $5000
01140            5000          START   EQU    *
01150                          *
01160  5000 9E   D9                    LDX    $D9     save POINTER
01170  5002 BF   5086                  STX    TEMP
01180  5005 8E   5055          NEXT    LDX    #PROMP1 output message
01190  5008 8D   41    504B            BSR    MESSAG
01200  500A BD   D807                  JSR    LINEIN  get number
01210  500D 6D   01                    TST    1,X     if null then end of program
01220  500F 27   34    5045            BEQ    END
01230  5011 9F   D9                    STX    $D9
01240  5013 BD   B50B                  JSR    STRFAC  string=>FAC1
01250  5016 BD   93E8                  JSR    FACPSH  FAC1=>stack
01260                          *
01270  5019 8E   505B                  LDX    #PROMP2
01280  501C 8D   2D    504B            BSR    MESSAG
01290  501E BD   D807                  JSR    LINEIN  get number
01300  5021 9F   D9                    STX    $D9
01310  5023 BD   B50B                  JSR    STRFAC  string=>FAC1
01320  5026 BD   9411                  JSR    FACPUL  stack=>FAC2
01330                          *
01340  5029 BD   934D                  JSR    FACSET
01350  502C BD   BE44                  JSR    FACCMP  FAC1:FAC2(Y-X)
01360                          *
01370  502F 5D                         TSTB
01380  5030 27   07    5039            BEQ    EQ      if X=Y
01390  5032 2B   0A    503E            BMI    LESS    if Y-X<0
01400  5034 8E   505F                  LDX    #MESGT  if Y-X>0
01410  5037 20   08    5041            BRA    MESOUT
01420  5039 8E   506C          EQ      LDX    #MESEQ
01430  503C 20   03    5041            BRA    MESOUT
01440  503E 8E   5079          LESS    LDX    #MESLES
01450  5041 8D   08    504B MESOUT     BSR    MESSAG
01460  5043 20   C0    5005            BRA    NEXT
01470                          *
01480  5045 BE   5086          END     LDX    TEMP    end
01490  5048 9F   D9                    STX    $D9
01500  504A 39                         RTS
01510                          *
01520                          * subroutine:output 1 line
01530                          *
01540  504B A6   80            MESSAG  LDA    ,X+
01550  504D 27   05    5054            BEQ    M01
01560  504F BD   D08E                  JSR    OUT
01570  5052 20   F7    504B            BRA    MESSAG
01580  5054 39                 M01     RTS
01590                          *
01600  5055      0D0A          PROMP1  FDB    $0D0A
01610  5057      58                    FCC    'X ='
01620  505A      00                    FCB    0
01630  505B      59            PROMP2  FCC    'Y ='
01640  505E      00                    FCB    0
01650  505F      20            MESGT   FCC    '    X < Y   !!'
01660  506B      00                    FCB    0
01670  506C      20            MESEQ   FCC    '    X = Y   !!'
01680  5078      00                    FCB    0
01690  5079      20            MESLES  FCC    '    X > Y   !!'
```

```
01700  5085       00                  FCB    0
01710                          *
01720  5086       0000         TEMP   FDB    0
01730                          *
01740             5000                END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5087
PROGRAM ENTRY ADDR=5000


exec &H5000

X =1.2345
Y =1.2346
   X < Y  !!
X =1.2345
Y =1.2345
   X = Y  !!
X =1.2345
Y =1.2344
   X > Y  !!
X =

Ready
```

単項演算

◀ＦＡＣの符号を反転する▶

機　能	ＦＡＣ１の符号を反転します.
パラメータ	ＦＡＣ１←数値 ＄１７←数値の型
アドレス	＄ＢＤＣ９
レジスタ	ＣＣ，Ａ，Ｂ，Ｘ，Ｕ
復帰情報	ＦＡＣ１：結果 ＄１７←結果の数値の型

解　説　ＦＡＣ１＝－ＦＡＣ１　を実行します.

サンプル

```
PAGE   001  (          ,          )

01000                           *
01010                           * sign change
01020                           *
01030                                   OPT     M,NOS,NOG,P=255
01040                 D807      LINEIN  EQU     $D807     line input
01050                 D08E      OUT     EQU     $D08E     output 1 character
01060                 D072      IN      EQU     $D072     input 1 character
01070                 B50B      STRFAC  EQU     $B50B     convert string into FAC1
01080                 B622      FACSTR  EQU     $B622     convert FAC1 into string
01090                 BDC9      SIGNC   EQU     $BDC9
01130  5000                             ORG     $5000
01140                 5000      START   EQU     *
01150                           *
01160  5000 9E        D9                LDX     $D9       save POINTER
01170  5002 BF        5052              STX     TEMP
01180  5005 8E        5035      NEXT    LDX     #PROMPT   output message
01190  5008 8D        21   502B         BSR     MESSAG
01200  500A BD        D807              JSR     LINEIN    get number
01210  500D 6D        01                TST     1,X       if null then end of program
01220  500F 27        14   5025         BEQ     END
01230  5011 9F        D9                STX     $D9
01240  5013 BD        B50B              JSR     STRFAC    string=>FAC1
01250  5016 BD        BDC9              JSR     SIGNC     FAC1<= - FAC1
01260  5019 8E        5044              LDX     #ANSWER
01270  501C 8D        0D   502B         BSR     MESSAG
01280  501E BD        B622              JSR     FACSTR    output FAC1
01290  5021 8D        08   502B         BSR     MESSAG
01300  5023 20        E0   5005         BRA     NEXT
01470                           *
01480  5025 BE        5052      END     LDX     TEMP      end
01490  5028 9F        D9                STX     $D9
01500  502A 39                          RTS
01510                           *
01520                           * subroutine:output 1 line
01530                           *
01540  502B A6        80        MESSAG  LDA     ,X+
01550  502D 27        05   5034         BEQ     M01
01560  502F BD        D08E              JSR     OUT
01570  5032 20        F7   502B         BRA     MESSAG
01580  5034 39                  M01     RTS
01590                           *
```

```
01600  5035      0D0A      PROMPT FDB    $0D0A
01610  5037      20               FCC    ' number X ? '
01620  5043      00               FCB    0
01630  5044      0D0A      ANSWER FDB    $0D0A
01640  5046      20               FCC    '       - X   ='
01650  5051      00               FCB    0
01710                      *
01720  5052      0000      TEMP   FDB    0
01730                      *
01740            5000             END    START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5053
PROGRAM ENTRY ADDR=5000


exec &H5000

 number X ? 123456789

      - X   =-123456789
 number X ? 1.23456789

      - X   =-1.23456789
 number X ? -1000000

      - X   = 1E+06
 number X ? 321654987654321354678945645132

      - X   =-3.216549876543214D+29
 number X ?

Ready
```

数値関数

機　能　FAC1を引数として関数値を求めます．

パラメータ　FAC1←数値（単精度実数）　$17←4（単精度実数）

アドレス
絶対値（ABS）――――――――――――$B3D5
アークタンジェント（ATN）――――――$BEF5
コサイン（COS）――――――――――――$BE60
eを底とする指数関数（EXP）――――――$BB5B
整数部分（FIX）――――――――――――$B461
引数の値を超えない最大の整数（INT）―$B472
自然対数（LOG）――――――――――――$B0AA
符号（SGN）――――――――――――――$B3BE
サイン（SIN）―――――――――――――$BE66
平方根（SQR）―――――――――――――$BAEE
タンジェント（TAN）――――――――――$BEB1

レジスタ　CC，A，B，X，U

復帰情報　FAC1：結果
$17：結果の数値の値

解　説　FAC1＝f（FAC1）を実行します．引数は単精度実数に限られ，演算も単精度で行われます（結果は実数であるとは限りません）．

サンプル

```
PAGE  001 (        ,        )

01000                            *
01010                            * function calculator
01020                            *
01030                                   OPT    M,NOS,NOG,P=255
01040                D807        LINEIN EQU    $D807    line input
01050                9BDB        MESSAG EQU    $9BDB    output from (X+1) till 0
01060                D08E        OUT    EQU    $D08E    output 1 character
01070                B337        FAC1.X EQU    $B337    FAC1=>(X)
01080                B301        FACX.1 EQU    $B301    (X)=>FAC1
01090                B50B        STRFAC EQU    $B50B    convert string into FAC1
01100                B620        FACSTR EQU    $B620    convert FAC1 into string
01110                BCE0        CSNG   EQU    $BCE0    convert into single procision
01120  5000                             ORG    $5000
01130                5000        START  EQU    *
01140                            *
01150  5000 9E       D9                 LDX    $D9      save POINTER
01160  5002 BF       50CA               STX    TEMP
01170  5005 8E       50BA        NEXT   LDX    #PROMPT-1 output message
01180  5008 BD       9BDB               JSR    MESSAG
01190  500B BD       D807               JSR    LINEIN
01200  500E 6D       01                 TST    1,X      if null then end of program
01210  5010 27       6C    507E         BEQ    END
```

```
01220  5012 CE   5084              LDU    #TABLE   get table addr.
01230  5015 A6   01         NO1    LDA    1,X      compare
01240  5017 A1   C4                CMPA   ,U
01250  5019 26   5B    5076        BNE    NO3
01260  501B A6   02                LDA    2,X
01270  501D A1   41                CMPA   1,U
01280  501F 26   55    5076        BNE    NO3
01290  5021 A6   03                LDA    3,X
01300  5023 A1   42                CMPA   2,U
01310  5025 26   4F    5076        BNE    NO3
01320                       *
01330  5027 5F                     CLRB            display function name
01340  5028 A6   C5         NO2    LDA    B,U
01350  502A BD   D08E              JSR    OUT
01360  502D 5C                     INCB
01370  502E C1   02                CMPB   #2
01380  5030 23   F6    5028        BLS    NO2
01390  5032 86   28                LDA    #'(
01400  5034 BD   D08E              JSR    OUT
01410  5037 30   03                LEAX   3,X      skip function name
01420  5039 9F   D9                STX    $D9      set pointer
01430  503B AE   43                LDX    3,U      get address
01440  503D BF   50CC              STX    ADDR
01450  5040 BD   B50B              JSR    STRFAC   string=>FAC1
01460  5043 BD   BCE0              JSR    CSNG     convert into SNG
01470  5046 8E   50CE              LDX    #FACX    save FAC1
01480  5049 BD   B337              JSR    FAC1.X
01490  504C BD   B620              JSR    FACSTR   FAC1=>string
01500  504F 30   1F                LEAX   -1,X
01510  5051 BD   9BDB              JSR    MESSAG   output
01520  5054 86   29                LDA    #')
01530  5056 BD   D08E              JSR    OUT
01540  5059 86   3D                LDA    #'=
01550  505B BD   D08E              JSR    OUT
01560  505E 86   04                LDA    #4       come back FAC1
01570  5060 97   17                STA    $17
01580  5062 8E   50CE              LDX    #FACX
01590  5065 BD   B301              JSR    FACX.1
01600  5068 AD   9F 50CC           JSR    [ADDR]   execute
01610  506C BD   B620              JSR    FACSTR   FAC1=>string
01620  506F 30   1F                LEAX   -1,X
01630  5071 BD   9BDB              JSR    MESSAG   output
01640  5074 20   8F    5005        BRA    NEXT
01650                       *
01660  5076 33   45         NO3    LEAU   5,U
01670  5078 1183 50BB              CMPU   #.TABLE
01680  507C 26   97    5015        BNE    NO1
01690                       *
01700  507E BE   50CA       END    LDX    TEMP     end of execution
01710  5081 9F   D9                STX    $D9
01720  5083 39                     RTS
01730                       *
01740                       * jump table
01750                       *      function name('XXX',3bytes)
01760                       *      function address (2bytes)
01770                       *
01780  5084      41         TABLE  FCC    'ABS'
01790  5087      B3D5              FDB    $B3D5
01800  5089      41                FCC    'ATN'
01810  508C      BEF5              FDB    $BEF5
01820  508E      43                FCC    'COS'
01830  5091      BE60              FDB    $BE60
01840  5093      45                FCC    'EXP'
01850  5096      BB5B              FDB    $BB5B
01860  5098      46                FCC    'FIX'
01870  509B      B461              FDB    $B461
01880  509D      49                FCC    'INT'
01890  50A0      B472              FDB    $B472
01900  50A2      4C                FCC    'LOG'
01910  50A5      B0AA              FDB    $B0AA
01920  50A7      53                FCC    'SGN'
```

```
01930  50AA      B3BE                FDB     $B3BE
01940  50AC      53                  FCC     'SIN'
01950  50AF      BE66                FDB     $BE66
01960  50B1      53                  FCC     'SQR'
01970  50B4      BAEE                FDB     $BAEE
01980  50B6      54                  FCC     'TAN'
01990  50B9      BEB1                FDB     $BEB1
02000            50BB      .TABLE EQU     *
02010                      *
02020  50BB      0D0A      PROMPT FDB     $0D0A
02030  50BD      66                FCC     'fff number ='
02040  50C9      00                FCB     0
02050                      *
02060  50CA      0000      TEMP   FDB     0
02070  50CC      0000      ADDR   FDB     0
02080  50CE      0004      FACX   RMB     4
02090                      *
02100            5000             END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50D1
PROGRAM ENTRY ADDR=5000

EXEC &H5000

fff number =SGN -5
SGN(-5)=-1
fff number =TAN 1
TAN( 1)= 1.55741
fff number =ATN 1.55741
ATN( 1.55741)= 1
fff number =FIX -1.5
FIX(-1.5)=-1
fff number =INT -1.5
INT(-1.5)=-2
fff number =EXP 2
EXP( 2)= 7.38906
fff number =LOG 7.38906
LOG( 7.38906)= 2
fff number =SQR 2
SQR( 2)= 1.41421
fff number =

Ready
```

5. I/O Address Mapping

本章では，ＦＭ７がサポートしている，Ｉ／Ｏのアドレスを解説しました．

各Ｉ／Ｏは，アドレス順にならべて解説してあります．

ＲＳ２３２Ｃとディスクについては，そのＩ／ＯをつかさどるＬＳＩの機能によるところが大きく，概略の記述にとどめました．詳しくは，各ＬＳＩの関係資料等を御参照下さい．

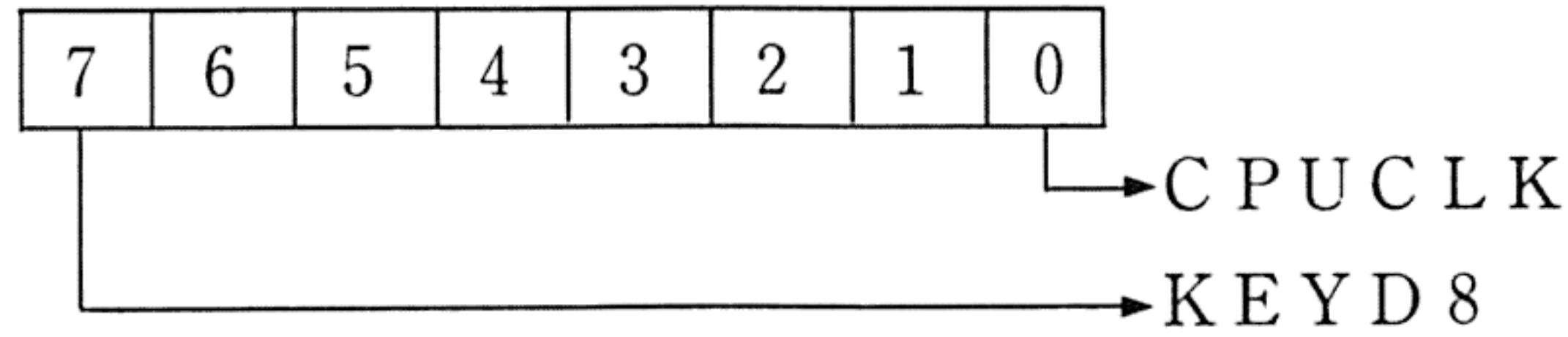
◀READ▶　このアドレスを読み出すときの機能を示します．

◀WRITE▶　このアドレスに書き込んだときの機能を示します．

$FD00

◀READ▶

7	6	5	4	3	2	1	0

- bit 0 → CPUCLK
- bit 7 → KEYD8

●CPUCLK　CPUの動作クロックを示します．動作クロックは後方のデップスイッチの4番によって決定されます．

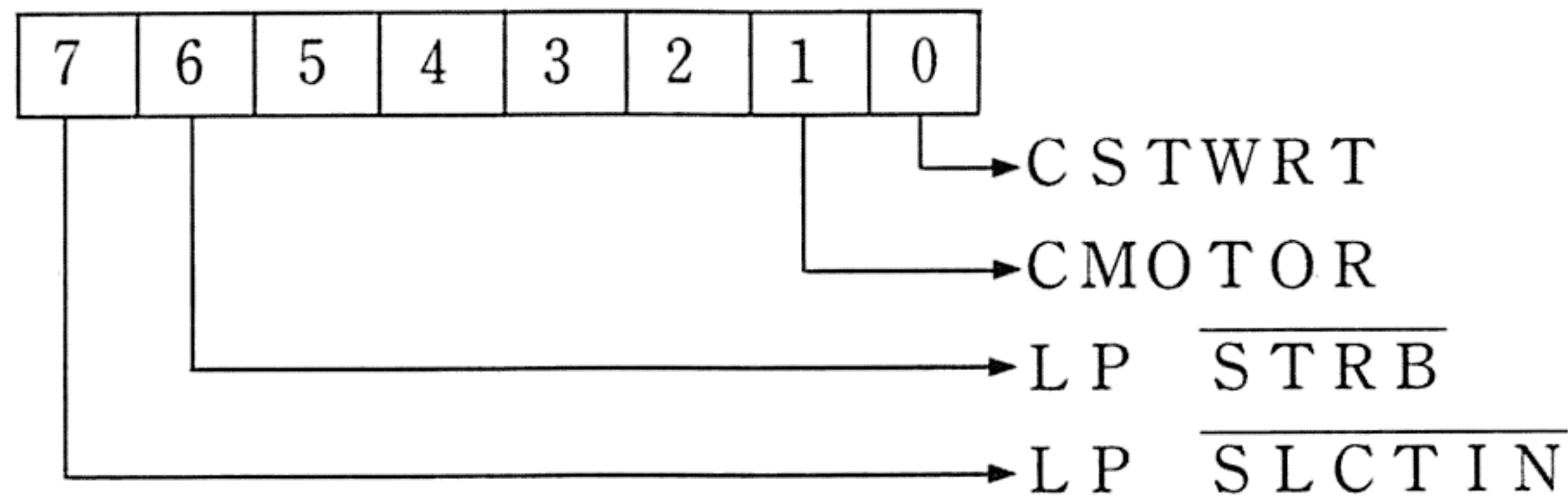
1：2MHz

0：1．2MHz

●KEYD8　キーボードデータの第8ビットです．キーボード関連については$FD01のReadの項を御参照下さい．

◀WRITE▶

7	6	5	4	3	2	1	0

- bit 0 → CSTWRT
- bit 1 → CMOTOR
- bit 6 → LP $\overline{\text{STRB}}$
- bit 7 → LP $\overline{\text{SLCTIN}}$

●CSTWRT　オーディオカセットへの出力データ

●CMOTOR　オーディオカセットのモータコントロール．

1：モータON

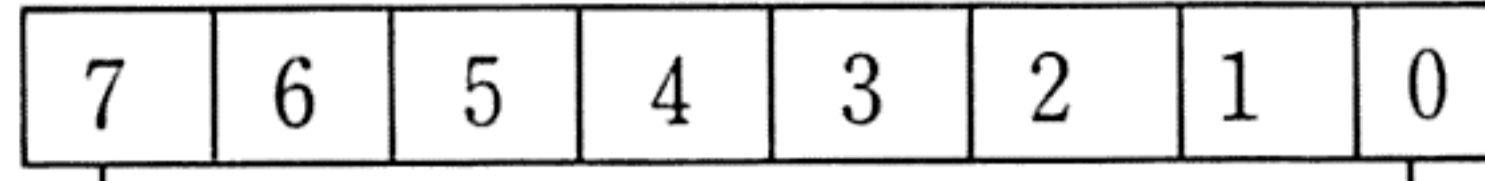
0：モータOFF

●LP $\overline{\text{STRB}}$　プリンタへストローブ信号を与えるためのものです．初期状態では1にセットしておき，$FD01に出力データをセットして，1→0→1と変化させることによってストローブ信号を出力します（プリンタ側のデータ読み込みは，立ち下がり（1→0）によって行われます）．

●LP $\overline{\text{SLCTIN}}$　プリンタ選択用の信号です．この信号を0にすると，プリンタが選択されて，オンライン状態になります．

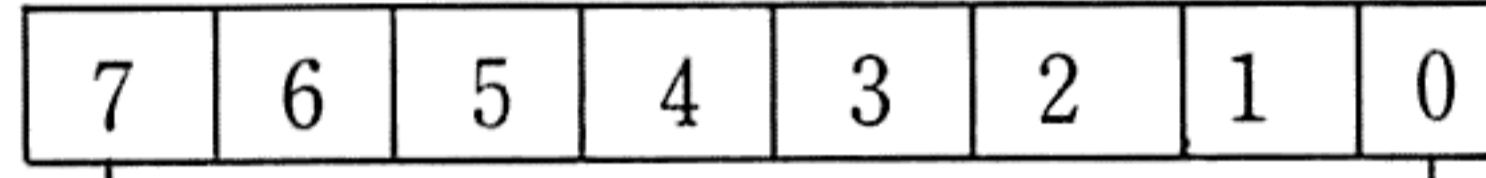
$FD01

◀READ▶

7	6	5	4	3	2	1	0

→KEYD0～KEYD7

●KEYD0
～
KEYD7　キーボードで最後に押されたキーのアスキーコードを示します．＄FD00のbit 7のKEYD8が1のときは，PFキーの番号を示します．

◀WRITE▶

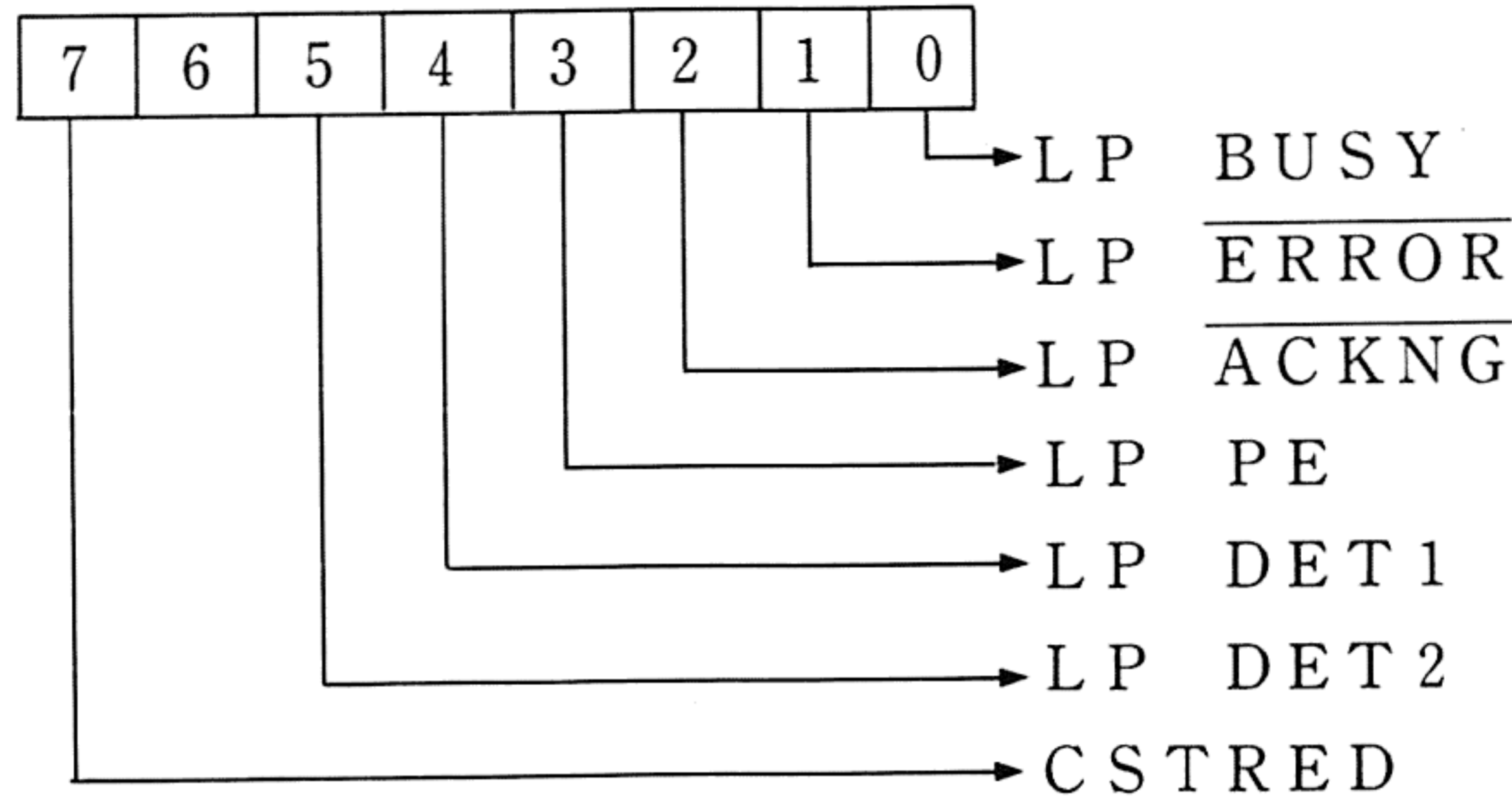
7	6	5	4	3	2	1	0

→PDATA0～PDATA7

●PDATA0
～
PDATA7　プリンタに出力するデータです．ここでデータを書き込んだ上で，$\overline{\text{LP STRB}}$信号を出力します．なおデータは，ストローブ信号（最低0．5μs）をはさんで前後最低0．5μsの間，保持される必要がありますが，2MHzの6809では最低（STA＄××）でも2μsを消費し，また，データはラッチされているので問題となりません．

＄FD02

◀READ▶

7	6	5	4	3	2	1	0

- 0 → LP BUSY
- 1 → LP $\overline{\text{ERROR}}$
- 2 → LP $\overline{\text{ACKNG}}$
- 3 → LP PE
- 4 → LP DET1
- 5 → LP DET2
- 7 → CSTRED

●LP BUSY　プリンタの状態を示す信号です．プリンタが動作可能（次のデータの出力が可能）の場合に0となります．プリンタが以下の状態の場合に1（BUSY）となります．

①データ読み取り中　　②印字中
③オフライン状態　　④エラー発生

プリンタにデータを送出するときには，このビジーフラグが0になるまで待つ必要があります．

●LP $\overline{\text{ERROR}}$　プリンタにエラーが発生した場合に0になります．このエラーフラグが0のときにはデータを送出するのは不適当です．

●LP $\overline{\text{ACKNG}}$　プリンタがデータの受け取りを終了し，データの保持を解除してもよいことを示す信号です．短期間（最低2 μs）の間，0になることによって伝えます．

本来は，プリンタ側の処理の終了は，この信号を用いて行われるべきですが，最低2 μsの信号を検出するのはCPUの処理スピードから困難であるので，通常のプリンタ出力では，LPBUSY信号をかわりに用います．

なお，この信号をCPUへのIRQ割り込みとして受け取ることもできます．（＄FD02 Write 参照）

●LP PE　プリンタが，紙なし状態になっていることを，信号が1であることによって示します．

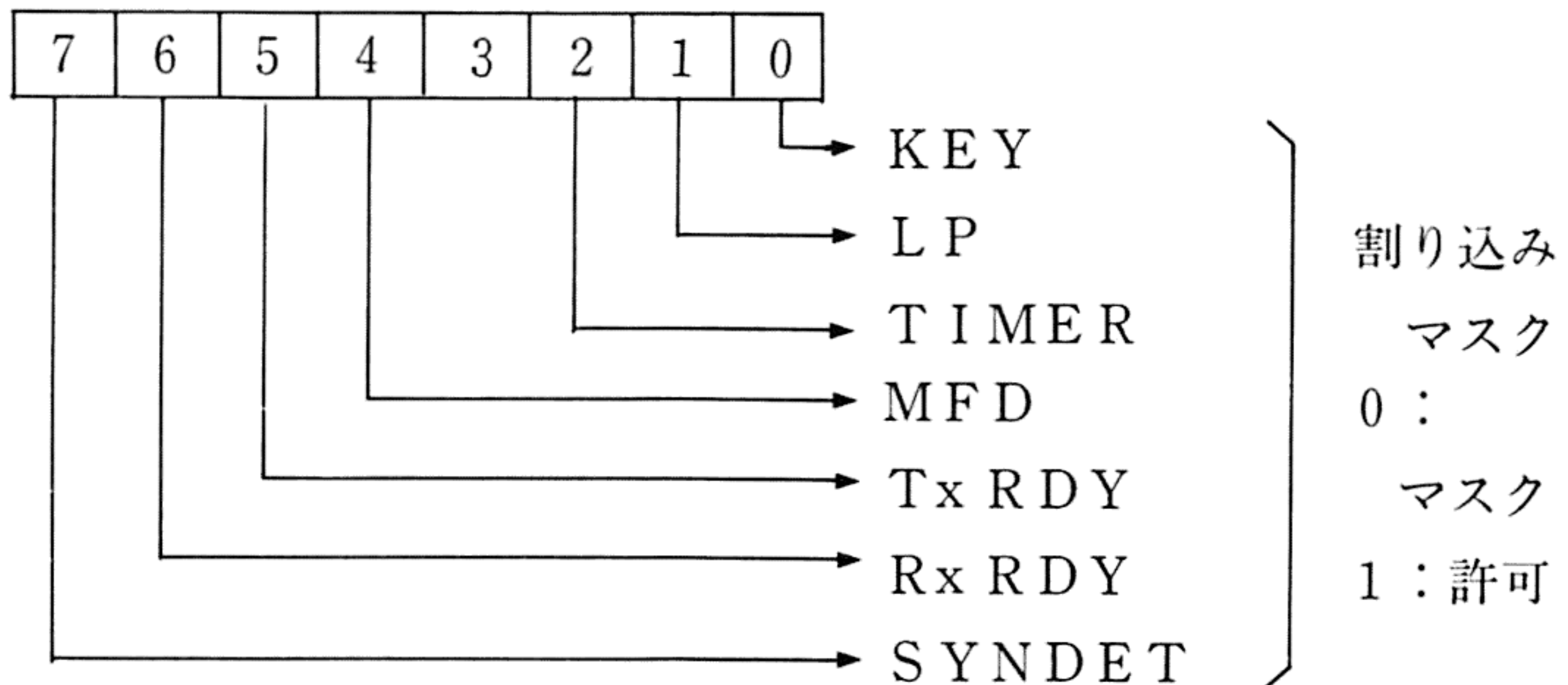
●LP DET1，2　接続されているプリンタの種類を検知するためのものです．それぞれは，プリンタコネクタの3番ピンと6番ピンの状態により決定されます．

●CSTRED　オーディオカセットからの入力データを示します．

◀WRITE▶

7	6	5	4	3	2	1	0

- 0 → KEY
- 1 → LP
- 2 → TIMER
- 4 → MFD
- 5 → TxRDY
- 6 → RxRDY
- 7 → SYNDET

割り込みマスク
0：マスク
1：許可

●KEY　このビットを１にすることにより，キーボードからの割り込み信号（キー入力があると発生するＩＲＱ）がメインＣＰＵに加わるようになります．また，それまでサブＣＰＵに加わっていたキーボードからの割り込み信号は加わらなくなります．

キーボードからの割り込み要求は，＄ＦＤ０１(キーデータ)をリードすることにより解除されます．

●LP　このビットを１にすることにより，プリンタが$\overline{\text{LPACKNG}}$信号を送ってきたときに，ＩＲＱが発生します．

この割り込み要求は，＄ＦＤ０３をリードすることにより解除されます．

●TIMER　このビットを１にすることにより，2．0３ msごとにＩＲＱが発生します．

この割り込み要求は，＄ＦＤ０３をリードすることにより解除されます．

●MFD　このビットを１にすることにより，ＦＤＣ（Floppy Disk Controller）からのＩＲＱがメインＣＰＵに加わるようになります．この割り込み要求は，＄ＦＤ１８をリードすることにより解除されます．

●TxRDY　ＵＳＡＲＴ（Universal Synchronous／Asynchronus Receiver／Transmitter：８２５１Ａ）を使用している場合，このビットを１にすれば，送信データを受けとれるようになるとＩＲＱが発生します．

この割り込み要求は，＄ＦＤ０６に送信データをセットすることにより解除されます．

●RxRDY　このビットを１にすることにより，ＵＳＡＲＴ使用時に，受信バッファにデータがセットされるとＩＲＱが発生します．

この割り込み要求は，＄ＦＤ０６にセットされている受信データを読み取ることにより，解除されます．

●SYNDET　このビットを１にすると，ＵＳＡＲＴが，同期式モードで内部Sync検出モードのときにSyncキャラクタを検出した場合か，非同期式モードでブレーク状態を検出した場合に，ＩＲＱを発生します．

以上のＩＲＱのうち，Ｆ－ＢＡＳＩＣは，Rx ＲＤＹとＴＩＭＥＲを使用しています．

Ｆ－ＢＡＳＩＣは，ＩＲＱが発生すると，要求元を判別し，それぞれ次のアドレスをサブルーチンコールするようになっています．このうち，ＴＩＭＥＲを除くそれぞれの番地にはＲＴＳ（＄３９）が初期設定されています．

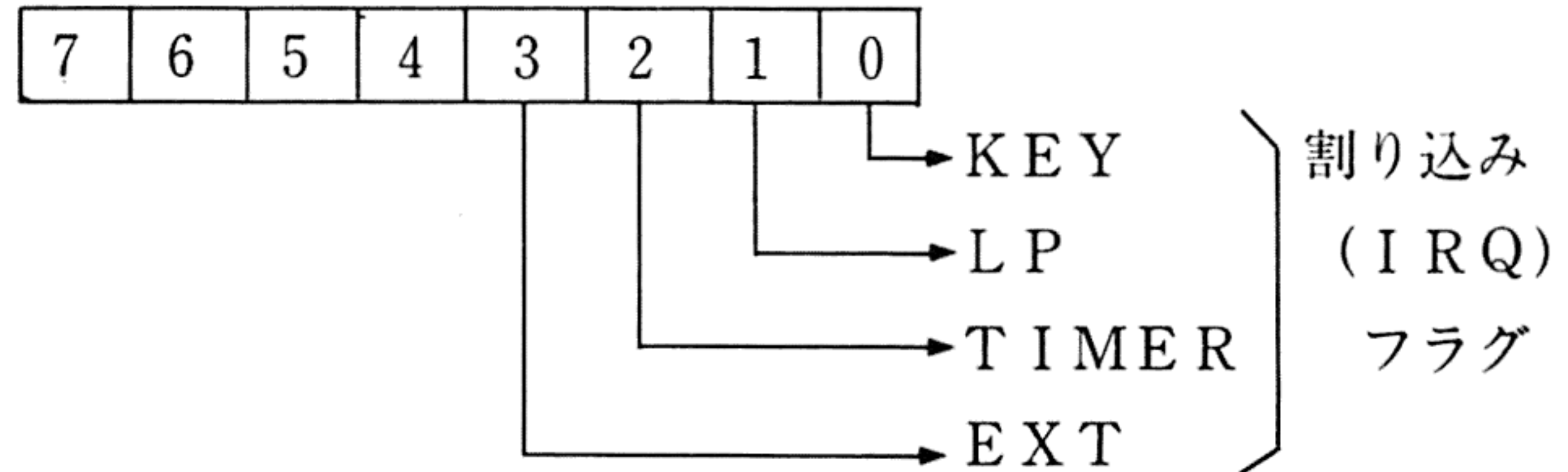
ＫＥＹ	＄０５Ｅ２
ＴＩＭＥＲ	＄０５ＤＦ
その他	＄０２９Ｃ

$FD03

◀READ▶

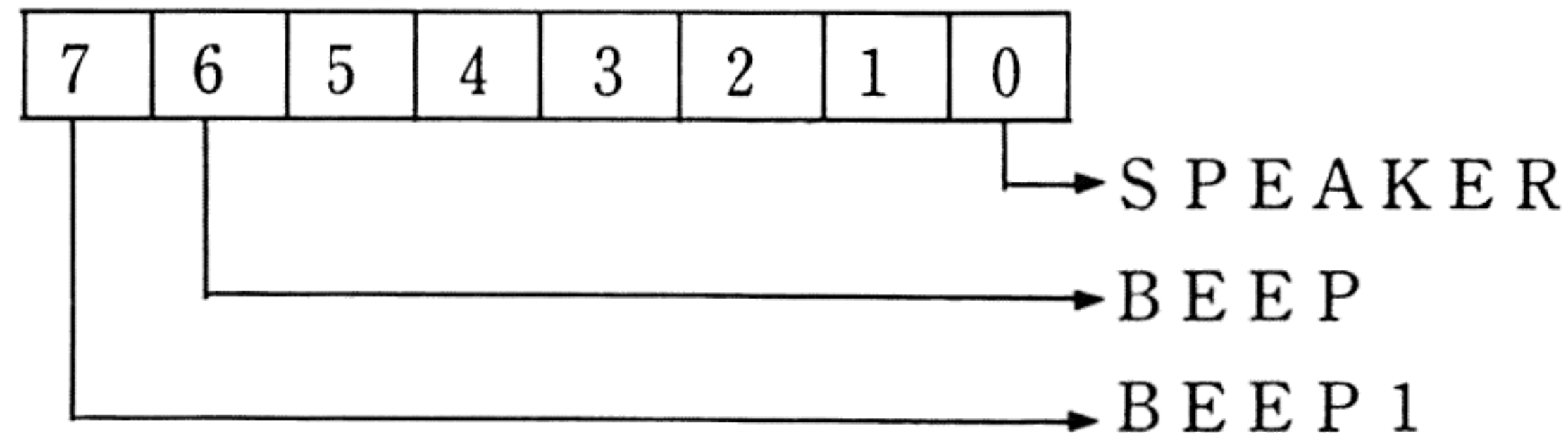
7	6	5	4	3	2	1	0

- bit 0 → KEY
- bit 1 → LP
- bit 2 → TIMER
- bit 3 → EXT

（KEY・LP・TIMER・EXT：割り込み（IRQ）フラグ）

ＩＲＱの要求元を調べるためのアドレスです．割り込み要求元のビットが0になります（ＥＸＴは，前出のＩＲＱのうち，ＳＹＮＤＥＴ，Rx ＲＤＹ，Tx ＲＤＹ，ＭＦＤのＩＲＱのときに0になります）．

◀WRITE▶

7	6	5	4	3	2	1	0

- bit 0 → SPEAKER
- bit 6 → BEEP
- bit 7 → BEEP1

●SPEAKER　スピーカーに音を出すかどうかを指定します．このビットが0の時は，ＢＥＥＰ，ＢＥＥＰ１の値によらず音は出ません．

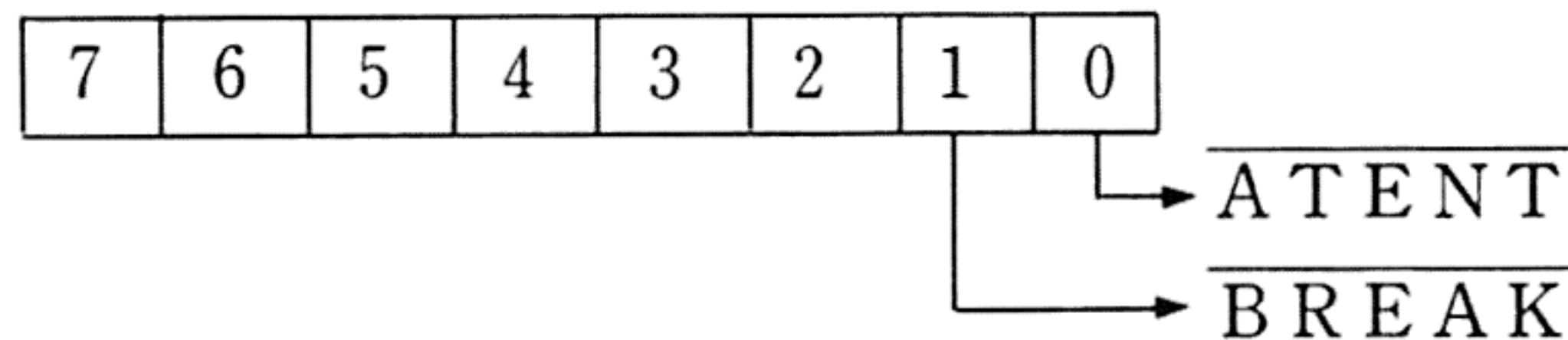
●BEEP　このビットに１を書き込むと，短時間ブザーがなります．

●BEEP1　このビットを１にすると，ブザーが鳴り続けます．このビットに0を書き込むと，ブザーは鳴り止みます．

$FD04

◀READ▶

7	6	5	4	3	2	1	0

- bit 0 → $\overline{ATENT}$
- bit 1 → $\overline{BREAK}$

ＦＩＲＱの要求元を示すアドレスで，割り込み要求元のビットが0になります．になります．

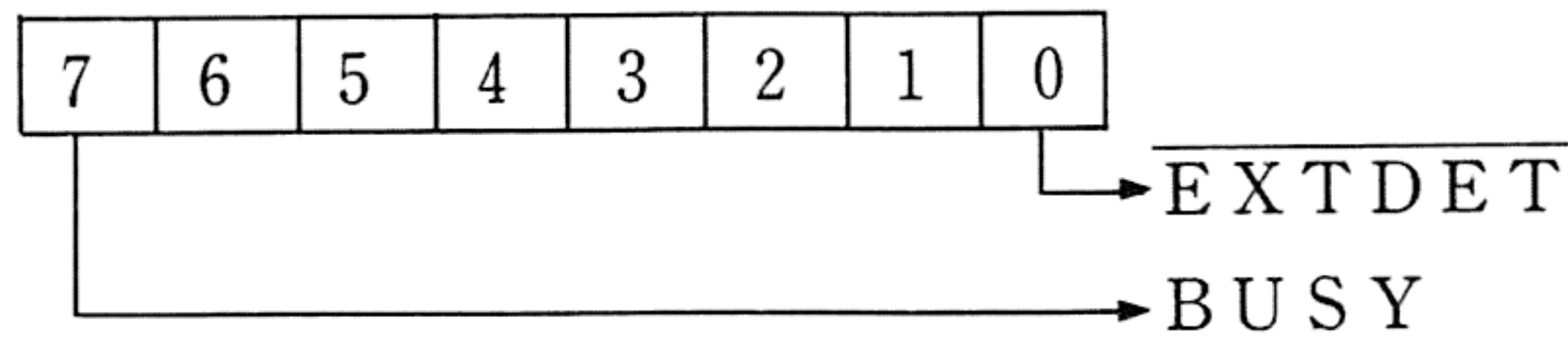
●$\overline{ATENT}$　サブＣＰＵからのＡＴＴＥＮＴＩＯＮ割り込み発生を示します．割り込み発生の要因としては，タイマ,ＰＦキーがあります．

●$\overline{BREAK}$　ＢＲＥＡＫキーが押されたことによって，割り込みが発生したことを示します．

$FD05

◀READ▶

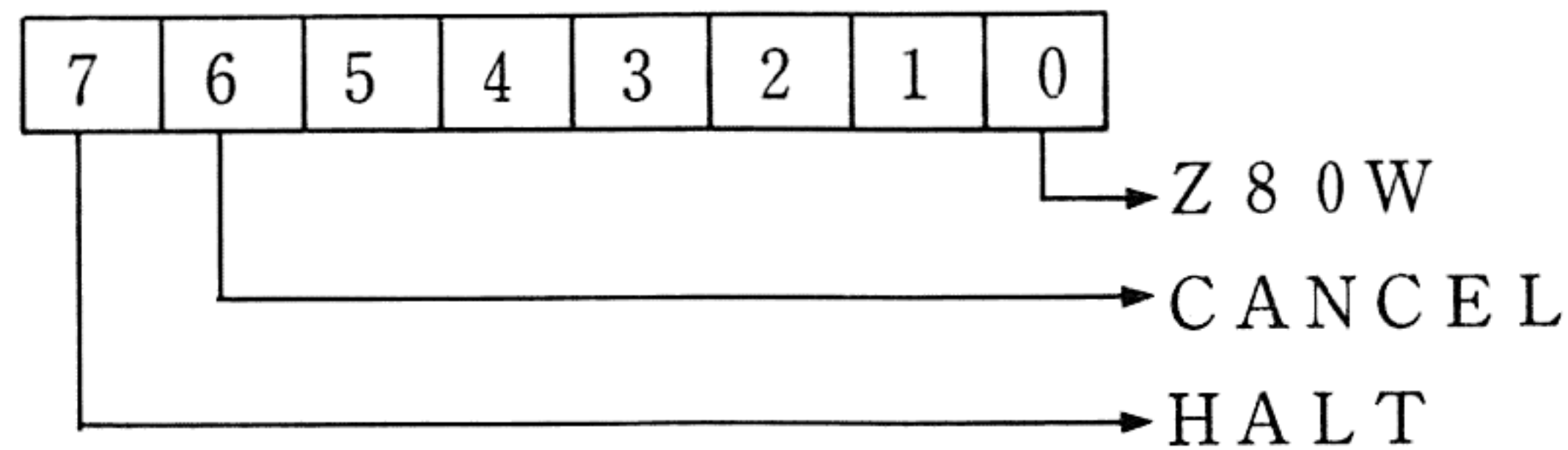
7	6	5	4	3	2	1	0

bit 0 → $\overline{\text{EXTDET}}$
bit 7 → BUSY

●$\overline{\text{EXTDET}}$　外部にディスク，RS232Cカードなどが拡張されている場合に0になります．

●BUSY　サブCPUがコマンド実行中であることを示します．

◀WRITE▶

7	6	5	4	3	2	1	0

bit 0 → Z80W
bit 6 → CANCEL
bit 7 → HALT

●Z80W　このビットに1を書き込むと，6809CPUはHALT状態になり，Z80ACPUのWAITが解除され，Z80ACPUが動作を開始します．
　また，Z80ACPUがこのビットに0を書き込むと，Z80ACPUはWAIT状態になり，6809CPUのHALTが解除され，6809CPUが動作を開始します．

●CANCEL　サブCPUがGET又は，グラフィックカーソルコマンド実行中にこのビットを1にすると，サブCPUはコマンドの実行を中止して，コマンド待ちになります．

●HALT　上記のBUSY信号が0（レディ）であることを確認した上で，このビットを1にするとサブCPUはHALT状態になり，共有RAMはメインCPUのアドレスバスに接続されます．HALTの解除は，このビットを0にすることによって行います．

$FD06

◀READ▶

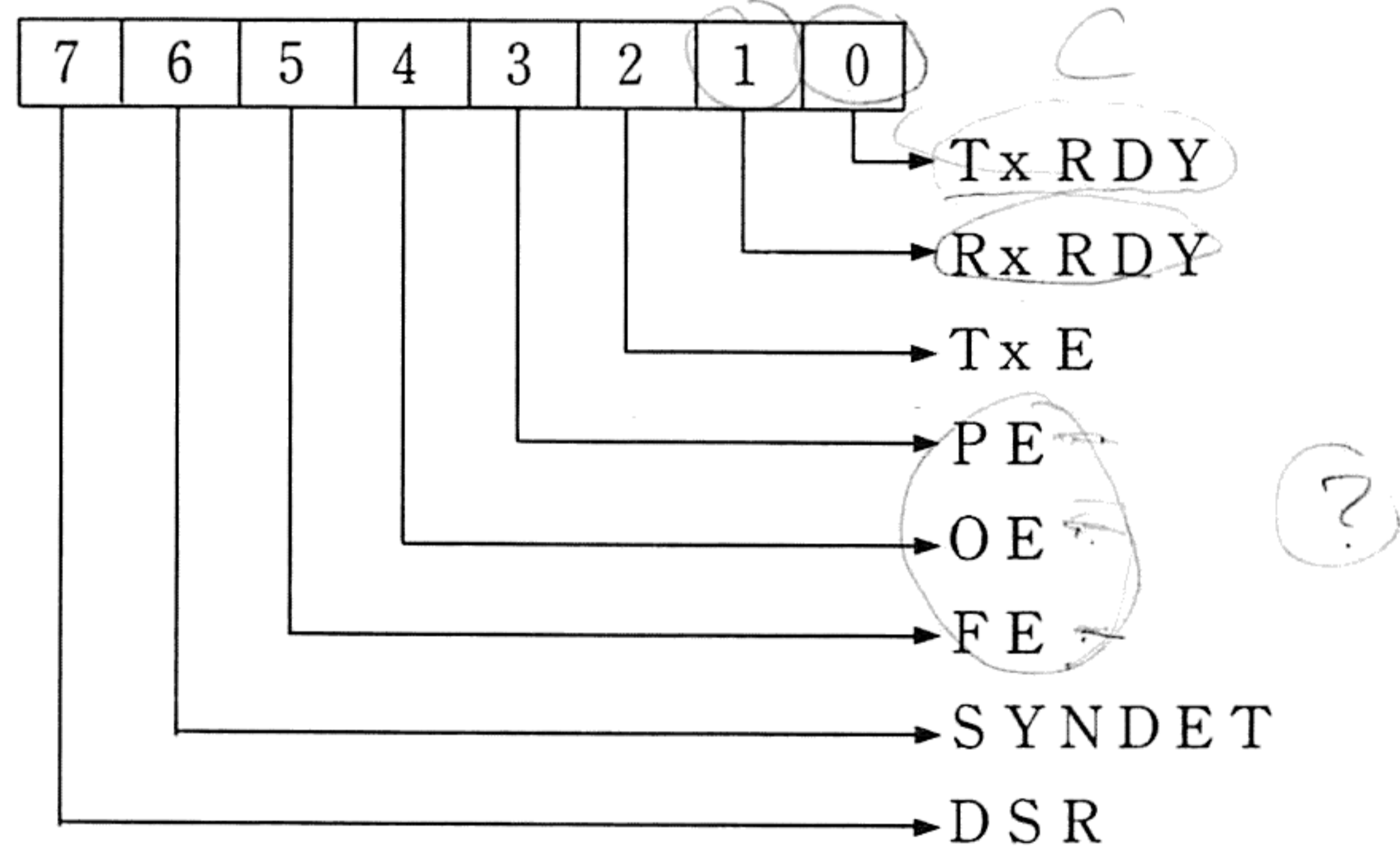
シリアル受信データ（USART8251Aの受信データレジスタ）

◀WRITE▶

シリアル送信データ（USART8251Aの送信データレジスタ）

$FD07

◀READ▶

7	6	5	4	3	2	1	0
DSR	SYNDET	FE	OE	PE	Tx E	Rx RDY	Tx RDY

USART8251Aのステータスレジスタです．

- **Tx RDY**　USARTがCPUからデータを受け取ることができるとき1となります．CPUからデータを受け取ると0となります．
- **Rx RDY**　USARTがCPUにデータを渡すことができる状態のときに1になります．CPUがデータを受け取ると自動的に0になります．
- **Tx E**　USARTが送信すべきデータを持たないときに1になります．CPUからデータを受けとると自動的に0になります．
- **PE**　パリティエラーが生じると1となります．
- **OE**　オーバーランエラーが生じると1となります．
- **FE**　フレーミングエラーが生じると1となります．
- **SYNDET**　同期式モードで内部Sync検出モードのときにSyncキャラクタを検出した場合か，非同期式モードでブレーク状態を検出した場合に1となります．

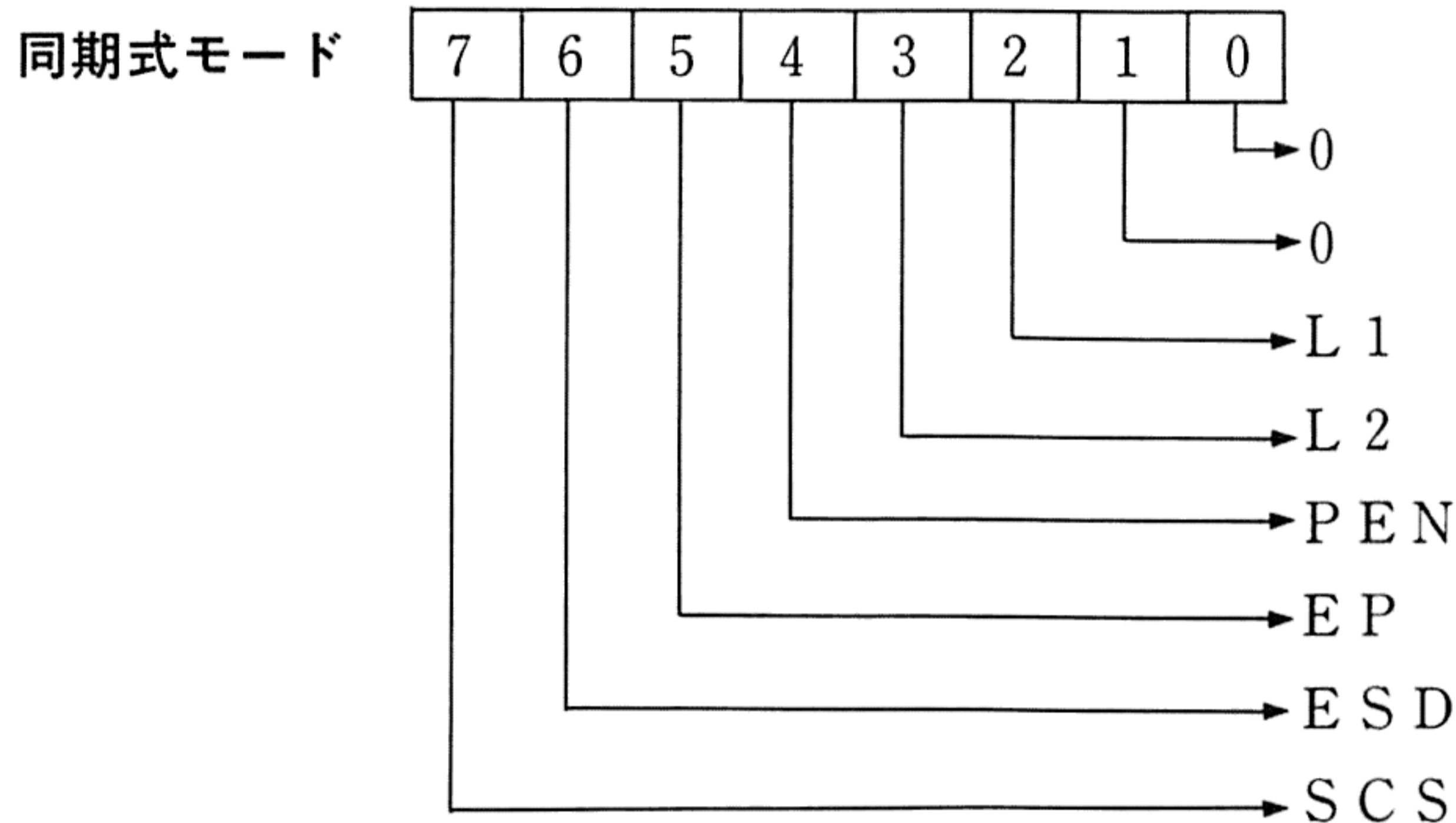
●DSR　　　USART8251Aの22番ピンの状態を反転したものを示します（このピンの用途は特に限定されていません）.

◀WRITE▶

USART8251Aのコントロールレジスタです．このレジスタの内容は，8251Aがリセット状態の直後であるか否かによって異なります．

◀モード・ワード▶（リセット状態の直後のとき）

同期式モード

7	6	5	4	3	2	1	0
SCS	ESD	EP	PEN	L2	L1	0	0

●L1，L2　　　1キャラクタの長さを示します．

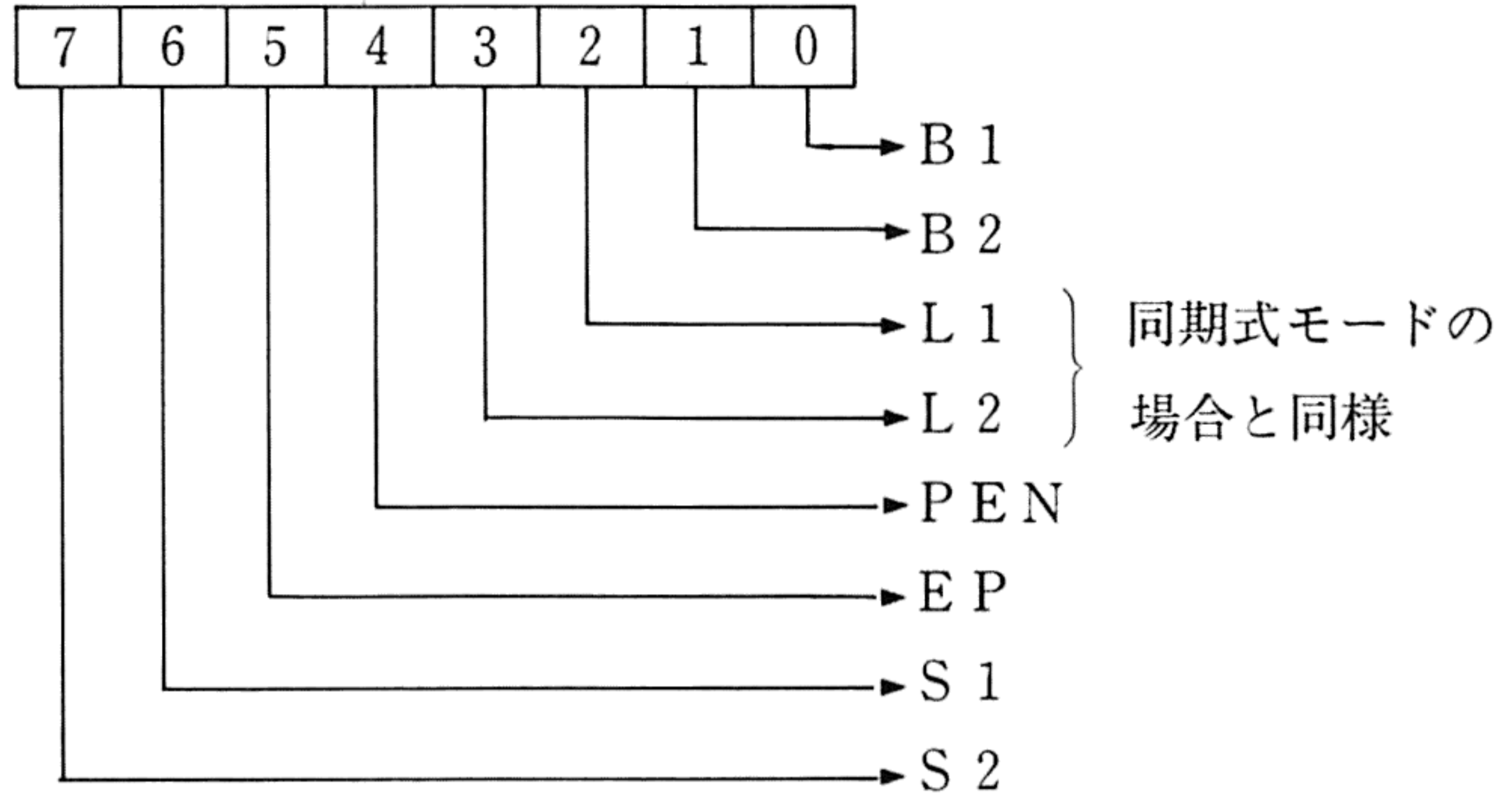
L1	0	1	0	1
L2	0	0	1	1
	5ビット	6ビット	7ビット	8ビット

●PEN　　　このビットが1であると，パリティ動作を行います．

●EP　　　パリティを指定します．1のとき偶数パリティ，0のとき奇数パリティになります．

●ESD　　　外部Sync検出モードを1で指定します．0のときには内部Sync検出モードとなります．

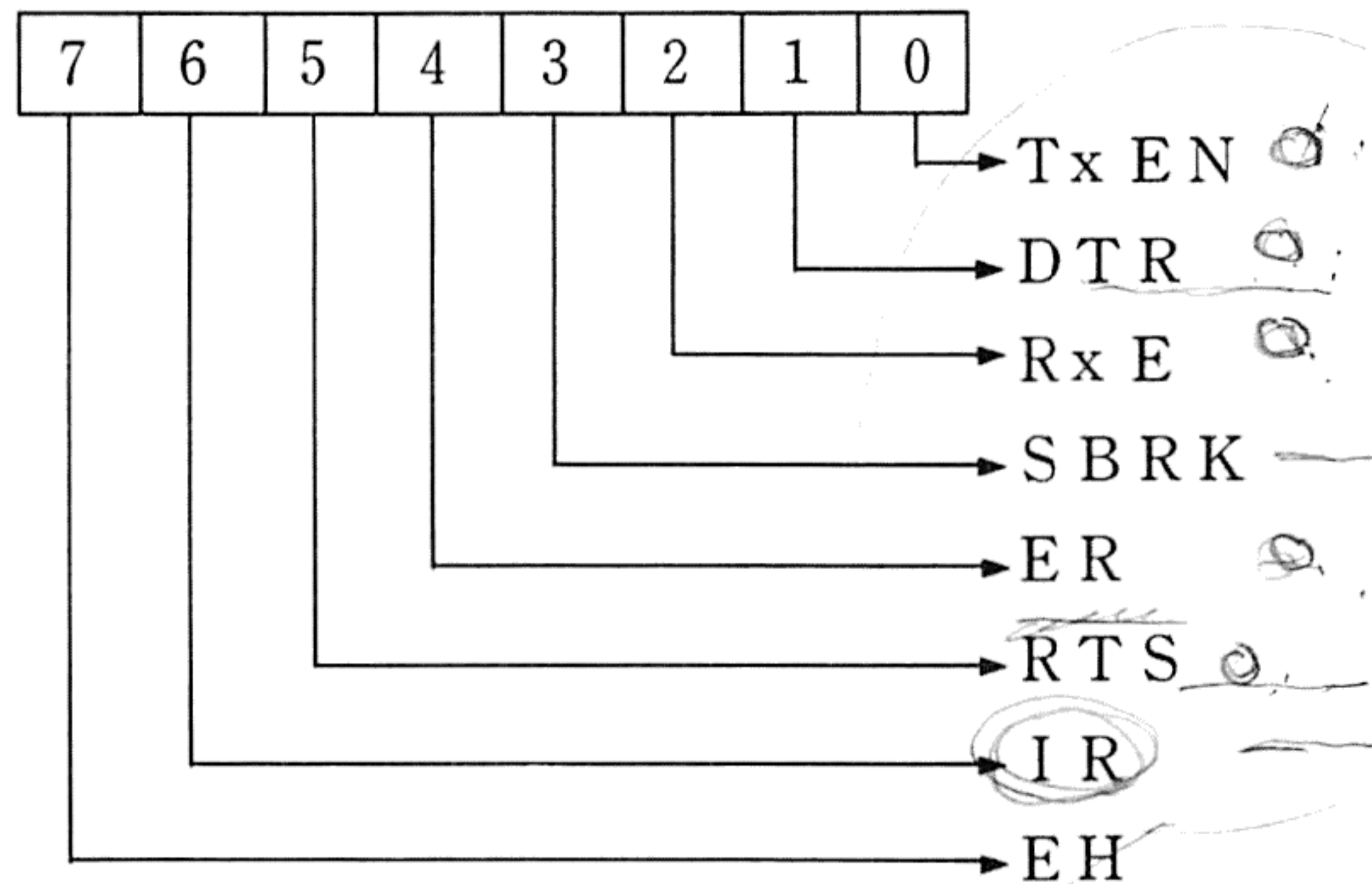
●SCS　　　1のときシングルキャラクタSyncモード，0のときダブルキャラクタSyncモードになります．

非同期式モード

●B 1, B 2　　ボーレートを指定します.

B 1	1	0	1
B 2	0	1	1
	×1	×16	×64

●S 1, S 2　　ストップビットの長さを指定します.

S 1	0	1	0	1
S 2	0	0	1	1
	無　　効	1 ビット	1½ビット	2 ビット

◀コマンド・ワード▶（リセット直後でない場合）

●Tx EN　　このビットを1にすることで送信動作が開始されます.

●DTR　　$\overline{\mathrm{DTR}}$（データ・ターミナル・レディ）端子をコントロールするビットです.

●RxE　　　このビットを１にすることで受信動作が開始されます．

●SBRK　　TxDの状態をこのビットを１にすることで強制的に〝L″にします．（センド・ブレーク・キャラクタ）

●ER　　　このビットを１にすると，ステータスレジスタのエラーフラグがリセットされます．

●RTS　　RTS（リクエスト・トゥ・センド）端子をコントロールするビットです．

●IR　　　８２５１Aをソフトウェアからリセットするためのビットで，１にセットされると８２５１Aはリセット状態になり，以後１回だけはモードワードを受け付けるようになります．

●EH　　　同期式モードの場合に，このビットを１にすると，ハントモードに入ります．

以上のRS２３２Cの詳細については，ｉ８２５１A関係資料を御参照下さい．

$FD0D

◀WRITE▶

PSGのコマンドレジスタです．以下にコマンドを示します．

＄００：インアクティブ・コマンド
＄０１：リード・コマンド
＄０２：ライト・コマンド
＄０３：レジスタ指定・コマンド

$FD0E

◀READ・WRITE▶

PSGのデータレジスタです．

例）レジスタ７にデータ＄０１を書き込む．

1．＄０７（レジスタ番号）を＄FD０Eに書き込む．
2．＄０３（レジスタ指定・コマンド）を＄FD０Dに書き込む．
3．＄００（インアクティブ・コマンド）を＄FD０Dに書き込む．
4．＄０１（書き込むデータ）を＄FD０Eに書き込む．
5．＄０２（ライト・コマンド）をFD０Dに書き込む．
6．＄００（インアクティブ・コマンド）を＄FD０Dに書き込む．

$FD0F

◀READ▶

ROMモードを選択します（$８０００～$ＦＢＦＦにROMを設定します）.

◀WRITE▶

RAMモードを選択します（$８０００～$ＦＢＦＦにRAMを設定します）.

注意）このモードの切換えを行うプログラムは，$００００～$７ＦＦＦに存在する必要があります.

$FD18

◀READ▶

ＦＤＣのステータスレジスタです．レジスタの内容は，ＦＤＣに与えるコマンドによって異なります.

◀WRITE▶

ＦＤＣのコマンドレジスタです．ＦＤＣのコマンドについては，富士通のＭＢ８８７７Ａ（又はその相当品）の関係資料を御参照下さい.

$FD19

◀READ・WRITE▶

ＦＤＣのトラックレジスタです.

$FD1A

◀READ・WRITE▶

ＦＤＣのセクタレジスタです.

$FD1B

◀READ・WRITE▶

ＦＤＣのデータレジスタです.

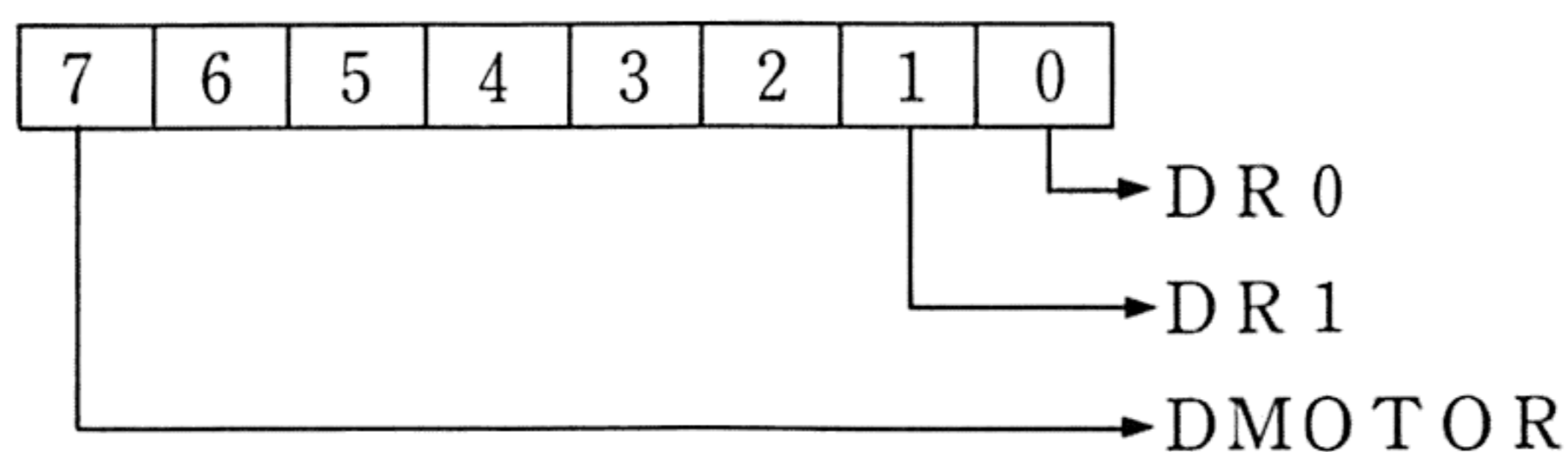
$FD1C

◀READ・WRITE▶

ディスクのサイドを指定します．最下位（LSB）が0のとき表，1のとき裏をアクセスすることを示します．

$FD1D

◀READ・WRITE▶

7	6	5	4	3	2	1	0

●DR0，DR1　アクセスするディスクドライブを指定します．

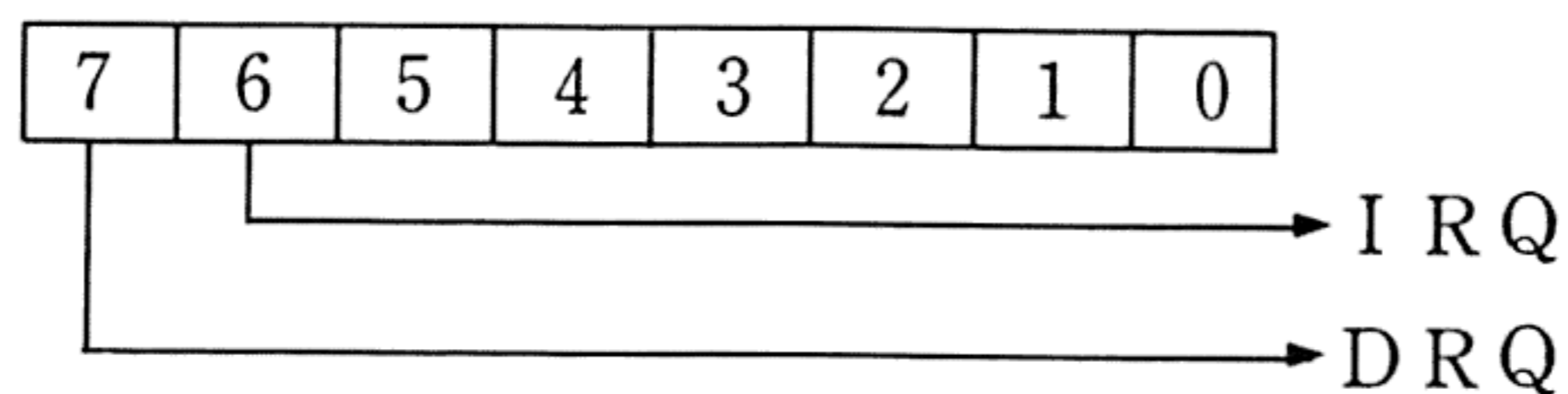
DR0	0	1	0	1
DR1	0	0	1	1
ドライブ番号	0	1	2	3

●DMOTOR　ディスクのモータを制御するためのビットです．1でモータが回転します．

Abortを行うと，ディスクのモータは停止してしまい，その後のディスクアクセスの際には，約7秒近く待たされることになります．これを避けるには，ディスクアクセスの前に，POKE&HFD1D，128を実行して，モータをあらかじめ回転させておくのが適当です．

$FD1F

◀READ▶

7	6	5	4	3	2	1	0

● IRQ　　FDCがコマンドを実行し終ると１になります．通常はメインCPUへの割り込みは行われませんが，割り込みを行なうようにも設定できます．（＄FD０２参照）

● DRQ　　データ要求が生じている場合に１となります．この要求があり次第CPUは，ただちに，データを受け取る（リード時）か書き込む（ライト時）かを行う必要があります．

＄FD２０，＄FD２１

◀WRITE▶

漢字ROMをアクセスする際に，ROMのアドレスを書きこみます．＄FD２０が上位，＄FD２１が下位になります．各漢字のROMアドレスは，「ユーザーズマニュアル・システム仕様」を御参照下さい．

＄FD２０，＄FD２１の上位１２ビットが各漢字を示し，下位４ビットは，そのビットパターンの位置を示します．

＄FD２２，＄FD２３

◀READ▶

上記の＄FD２０，＄FD２１で指定した漢字パターン１６ビットのうち，＄FD２２に左側の８ビット，＄FD２３に右側の８ビットのビットパターンが現われます．

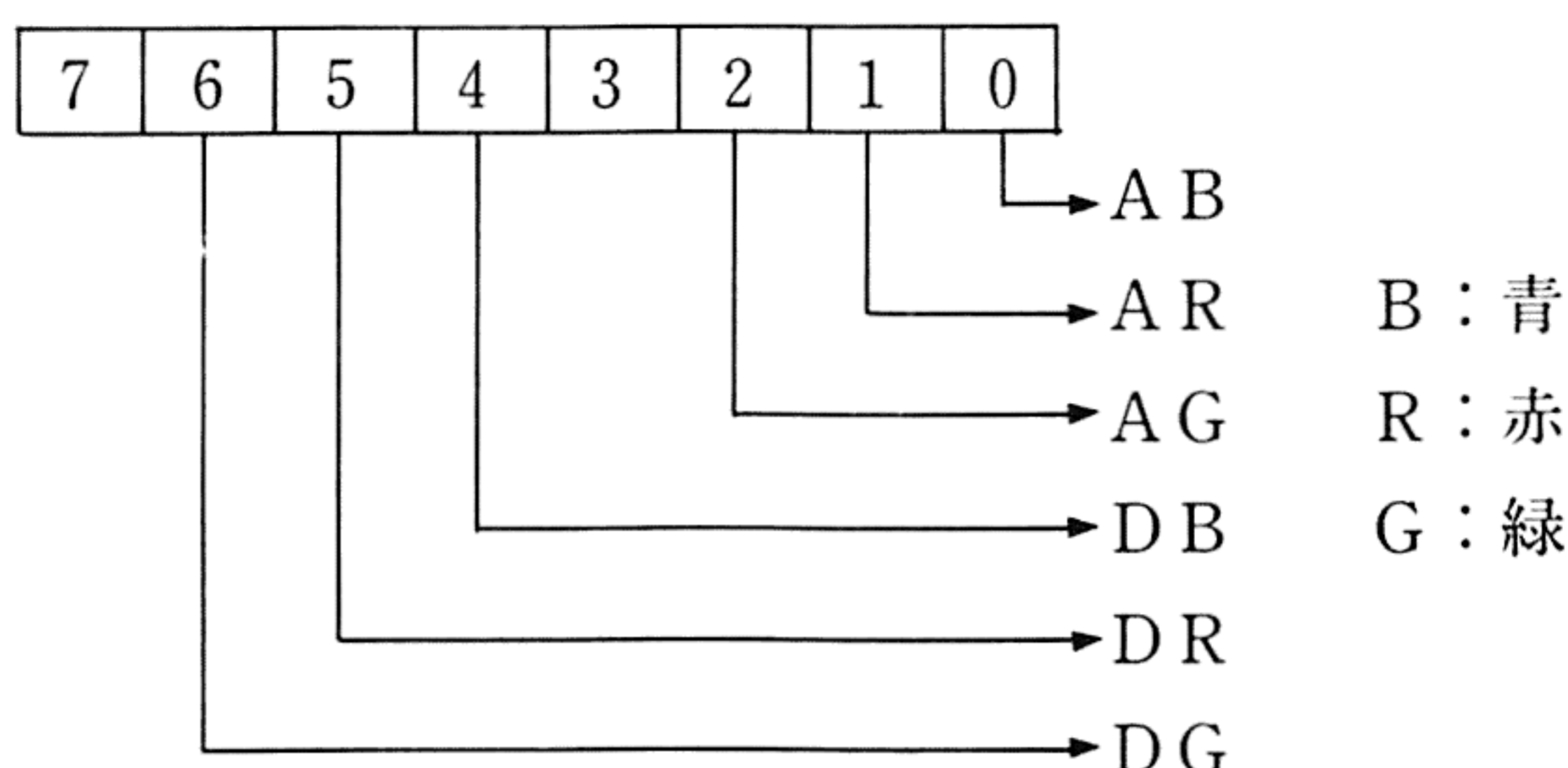
＄FD２４～＄FD２B

拡張ユニット内に増設される，RS２３２Cインタフェースモジュール用のアドレスです．それぞれの内容を以下に示します．

ディバイス名	COM１：	COM２：	COM３：	COM４：
データレジスタ＄FD０６と同じ	＄FD２４	＄FD２６	＄FD２８	＄FD２A
コマンドレジスタ＄FD０７と同じ	＄FD２５	＄FD２７	＄FD２９	＄FD２B

$FD37

◀WRITE▶

7	6	5	4	3	2	1	0

●AB，AR，AG

対応するビットを1とすると，そのVRAMに対してのサブCPUのアクセスは禁止されます．アクセスが禁止されると，サブCPUはそのVRAMへの書き込みができなくなる他，そのVRAMを読んでも，$FFが返されます．しかし，全画面がスクロール画面の場合のスクロールは実行されてしまうので注意が必要です．これは，FM7が全画面のスクロールを，この指定に関係のないハードウェアで行っているためです．

●DB，DR，DG

対応するビットが0のVRAMが画面に表示されます．

$FD38～$FD3F

パレットレジスタです．各レジスタとも，bit 0がB，bit 1がR，bit 2がGのカラーに対応しています．bit 3～bit 7は関係ありません．

◀READ・WRITE▶

	パレットコード	リセット時の色		パレットコード	リセット時の色
$FD38	0	黒	$FD3C	4	緑
$FD39	1	青	$FD3D	5	水色
$FD3A	2	赤	$FD3E	6	黄
$FD3B	3	マゼンダ	$FD3F	7	白

画面のボーダーカラー（CRTの帰線区間の色）は，パレットコード0となっているので，$FD38に黒以外のカラーコードを書き込むことにより，ボーダーカラーが指定できます．

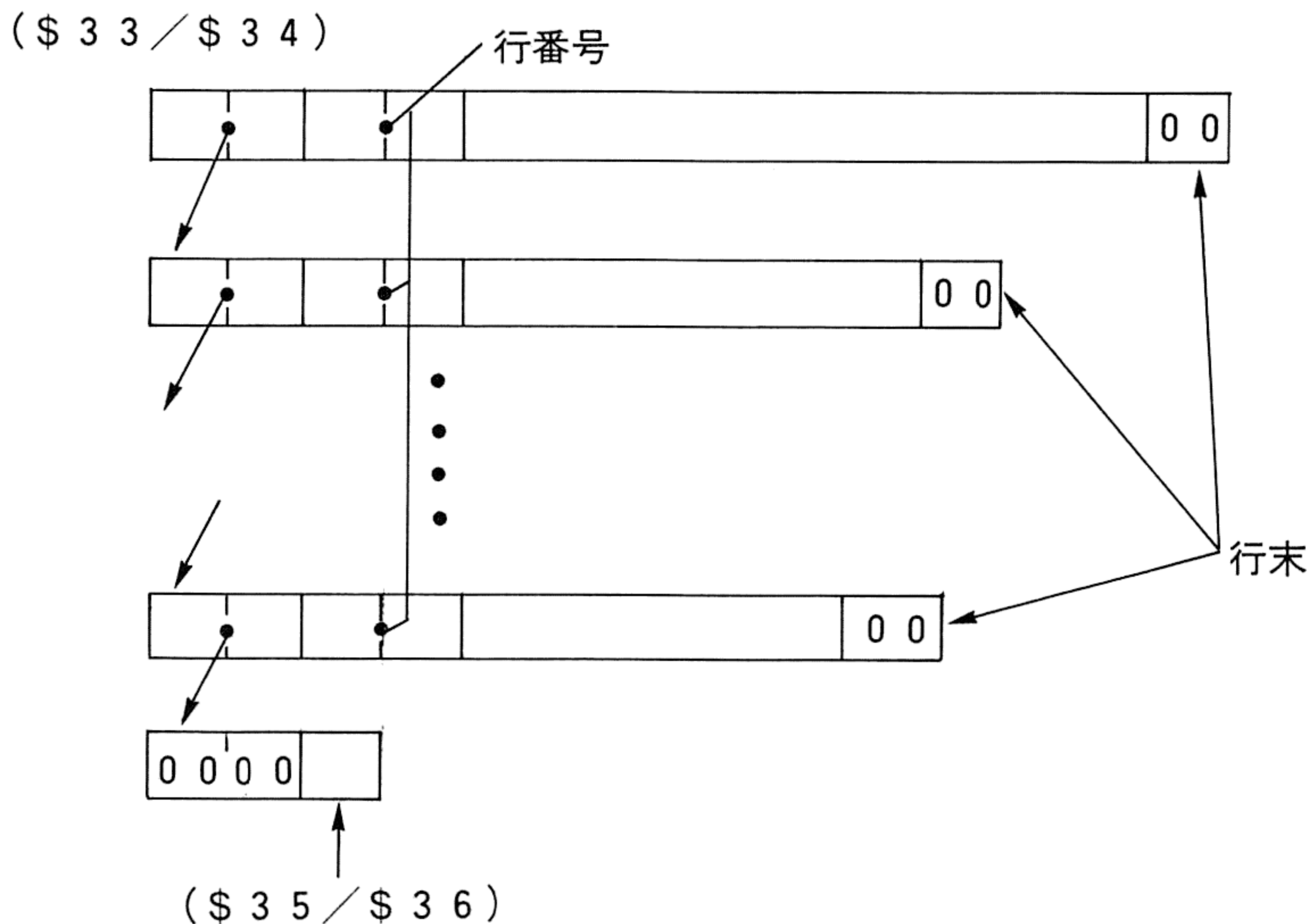
6. System Workarea

本章では，Ｆ－ＢＡＳＩＣのワークエリアについて解説します．

Ｆ－ＢＡＳＩＣは，＄１１～＄７８Ｆ（ＲＯＭモードの場合）をワークエリアとして，プログラムを実行しています．これらの中には，あるサブルーチンでしか使われないもの，また，いろいろなルーチンで別の用途に用いているものなどさまざまです．そこで，この章では，第３章，第４章で述べたルーチンに必要なものと，ユーザーが利用することの多いと思われるワークエリアを中心に解説しました．そのため，アドレスが飛び飛びになっていますが，その間のアドレスもいろいろな用途に使われていることに注意して下さい．また，本章では，特に指定する以外は，ＲＯＭモードである場合について書かれていますので注意して下さい．

アドレス	内　容
＄００１５	実数演算を行う時に，その演算を倍精度で行うか，単精度で行うかを示すフラグです． ０：単精度 ０以外：倍精度
＄００１７	ＦＡＣ１に格納されている数の型を示す値が入ります． ２：整数 ３：文字列 ４：単精度実数 ８：倍精度実数 この値はそのまま，ＦＡＣのうち何バイトを使用しているかを示す値になっています．
＄００２７～＄００２Ｆ	ＦＡＣ３として使用します．＄００２７が指数部，＄００２８～＄００２Ｅが仮数部，＄００２Ｆが符号部となります(実数の場合)．このＦＡＣは，乗算，除算などの時にワークとして使用します．
＄００３３／＄００３４	ＢＡＳＩＣのテキストの先頭アドレスを示します．ＲＯＭモードでは通常＄０７９０となります．この値を書き替えてからＮＥＷを実行することにより，ＢＡＳＩＣのテキストの格納アドレスを後方にずらすことが可能になります． 例）ＰＯＫＥ　＆Ｈ３３,＆Ｈ１０ 　　ＰＯＫＥ　＆Ｈ３４,＆Ｈ００ 　　ＮＥＷ これにより，ＢＡＳＩＣのテキストは，＄１０００以降に格納されるようになり，＄７９０～＄１０００を自由に使えます．
＄００３５／＄００３６	ＢＡＳＩＣのテキストの終端のアドレスを示します．ＮＥＷ直後の状態では，＄００３３／＄００３４の値に２を加えた値になっています．

◀ＢＡＳＩＣのテキストの構造▶

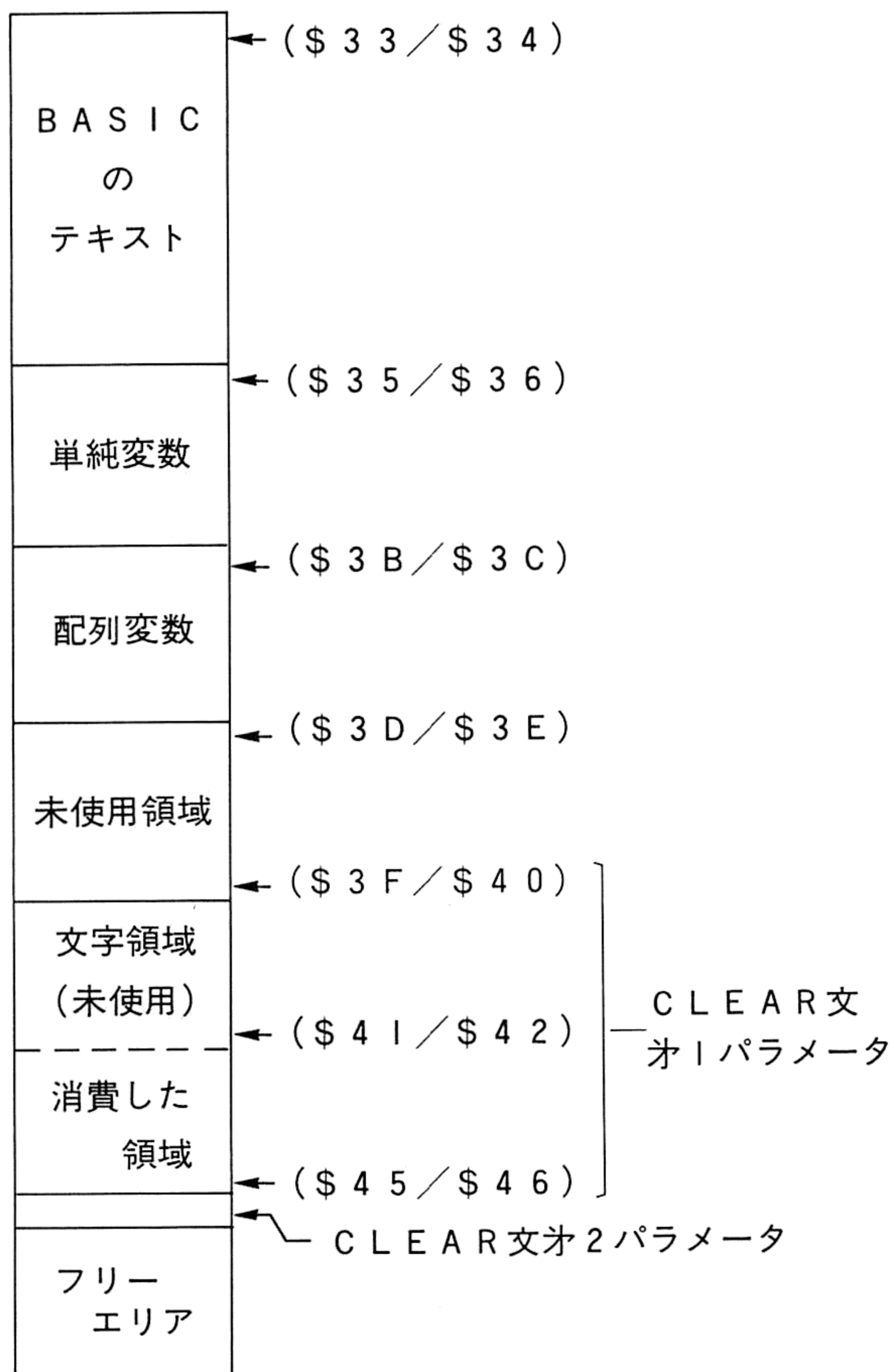
（＄３３／＄３４）
行番号
００
００
００
行末
０００００
（＄３５／＄３６）

また，この値は同時に単純変数の格納エリアの先頭アドレスも示しています．

＄００３Ｂ／＄００３Ｃ　配列変数の格納エリアの先頭アドレスを示します．

＄００３Ｄ／＄００３Ｅ　ＢＡＳＩＣの使用領域のうち，まだ未使用である部分の先頭アドレスを示します．

＄００３Ｆ／＄００４０　ＢＡＳＩＣの文字領域の先頭アドレスを示します．実際には，ＣＬＥＡＲ文の第２パラメータから第１パラメータの値を引いた値からさらに１引いた値です．

＄００４１／＄００４２　ＢＡＳＩＣの文字領域は，アドレスの上位から消費されていきますが，この値はすでに使用されている文字領域の先頭アドレス－１の値を示します．この値が＄００３Ｆ／＄００４０に近づけば，カベージコレクションが行われるということを示しています．

＄００４５／＄００４６　ＣＬＥＡＲ文の第２パラメータ（ＢＡＳＩＣの使用する上限）の値から１を引いた値を示します．

＄００４７／＄００４８	実行中の行の行番号を示します．ダイレクトモードの時には，＄ＦＦＦＦがセットされています．
＄００４９／＄００４Ａ	ＣＯＮＴで実行再開する行の行番号がはいります．
＄００４Ｄ／＄００４Ｅ	ＣＯＮＴで実行再開するステートメントのあるアドレスがはいります．この値が0のときには，Can't Continue エラーとなります．
＄００４Ｆ／＄００５０	実行中のステートメントのあるアドレスを示します．
＄００５１／＄００５２	読みとっているＤＡＴＡ文のある行番号を示します．
＄００５３／＄００５４	読みとっているＤＡＴＡのあるアドレスを示します．
＄００５Ｄ	ＦＡＣ２に格納されている数の型を示す値がはいります．値は，＄００１７のそれと同じです．しかし，この値は常に正確にセットされているとは限らないので注意が必要です．
＄００７４～＄００７Ｃ	ＦＡＣ１として使用します．くわしくは第４章を御参照下さい．
＄００８２～＄００８Ａ	ＦＡＣ２として使用します．
＄００９Ｄ	ＡＵＴＯ機能を選択するかどうかのフラグです． ０：ＡＵＴＯ機能は選択しない． ０以外：ＡＵＴＯ機能を選択する．
＄００９Ｅ／＄００９Ｆ	ＡＵＴＯ機能での次の行番号を示します．
＄００Ａ０／＄００Ａ１	ＡＵＴＯ機能での行番号の増分を示します．
＄００Ａ６／＄００Ａ７	ＬＩＳＴなどで用いられる，“．”（ピリオド）の行番号を示します．

＄００Ａ９	ＴＲＡＣＥ機能を選択するかのフラグです． ０：ＴＲＡＣＥ機能は選択しない ０以外：ＴＲＡＣＥ機能を選択する．
＄００ＡＣ	エラーコードがはいります．システム変数ＥＲＲを示します．
＄００ＡＤ／＄００ＡＥ	エラーがおきた行番号を示します．システム変数ＥＲＬを示します．ダイレクトモードでエラーが生じた場合には，＄ＦＦＦＦとなります．
＄００Ｂ１	通常は０であるが，ＯＮＥＲＲＯＲＧＯＴＯによるエラー処理ルーチン実行中は＄ＦＦとなります．
＄００Ｂ２／＄００Ｂ３	エラーをおこしたステートメントのあるアドレスを示します．
＄００Ｂ４／＄００Ｂ５	ＯＮＥＲＲＯＲＧＯＴＯで指定した行のあるアドレスを示します．この値が０のときは，エラートラップ機能は選択されません．
＄００Ｂ８～＄００ＢＡ	数値出力の際のフォーマット指定を示します．くわしくは，第４章を御参照下さい．
＄００ＢＦ	入出力の際に用いられる論理機番を示します．
＄００Ｃ０	入力の際にエラーが生じたかどうかを示すフラグです．０以外のときはエラーが生じたことを示します．生じるエラーには，Input Past End., Device I／O Error 等があります．

$００Ｃ３	画面の桁数（ＷＩＤＴＨ文第１パラメータ）を示します．
$００Ｄ１	プロテクトフラグです．この値が０以外であると，プロテクトがかかっていることを示します．プログラムにプロテクトがかかっている時には，モニタからこのアドレスを０にすることにより，プロテクトは解除できます．
$００ＤＥ～$００Ｅ０	ＢＩＯＳへのジャンプ命令（ＪＭＰ$Ｆ１７Ｄ）が書かれています．
$００ＥＢ／$００ＥＣ	最後に実行されたＬＩＮＥ文の終端のＸ座標を示します．
$００ＥＤ／$００ＥＥ	最後に実行されたＬＩＮＥ文の終端のＹ座標を示します．
$０１００～$０１７Ｆ	ＢＡＳＩＣ内部からDisplay Sub Systemを使用するときにコマンドをセットする領域です．
$０１Ｄ１～$０１Ｅ２	メインＣＰＵに割り込み（ＳＷＩをふくむ）がかかると，それぞれ以下のアドレスにジャンプしてきます．（$ＦＦＦ２～$ＦＦＦＣのベクトルがそれぞれ以下のアドレスを指している） 初期状態では，ＳＷＩ，ＳＷＩ２，ＳＷＩ３，ＮＭＩのアドレスにはＲＴＩ（$３Ｂ）が書かれており，ＩＲＱのアドレスには，ＪＭＰ$Ｄ２ＦＣ，ＦＩＲＱのアドレスには，ＪＭＰ$Ｃ９５３が書かれています． ユーザーがこれらの割り込みを使用する場合には，$ＦＦＦ２～$ＦＦＦＣのベクトルか，このアドレスを書き替えて，割り込み処理をすることになります．

ＳＷＩ３	＄１Ｄ１
ＳＷＩ２	＄１Ｄ４
ＳＷＩ	＄１Ｄ７
ＮＭＩ	＄１ＤＡ
ＩＲＱ	＄１ＤＤ
ＦＩＲＱ	＄１Ｅ０

＄０１Ｅ３　　ＬＰＴ０を改行する際に，マスクするコードを示します．

通常＄０Ｄがセットされています．ＬＰＴ０を改行する際には，＄０Ｄ，＄０Ａが順に出力されますが，そのうち，＄０Ｄはマスクされて出力されず，＄０Ａのみが出力されます．しかし，改行以外の時に＄０Ｄを出力した場合には，そのまま出力されます．

＄０１Ｅ５　　キャラクタの色を示します．

＄０１Ｅ６　　背景色を示します．

＄０１Ｅ７／＄０１Ｅ８　　ＵＮＬＩＳＴの行番号を示します．ＬＩＳＴ時に，この値以上の行は表示しません．これを解除する時には，このアドレスに＄ＦＦＦＦを書き込みます．

＄０１ＥＢ　　ディバイス名を指定しなかった場合に選択するデバイスの物理機番を示します．ＲＯＭモードのとき＄０２，ＤＩＳＫモードのとき＄８０となります．

＄０１ＥＦ　　基本のステートメントの数を示します．（＄６８）

＄０１Ｆ０／＄０１Ｆ１　　基本のステートメントの予約語テーブルのアドレスを示します．（＄８８９０）

＄０１Ｆ２／＄０１Ｆ３　　基本のステートメントのジャンプテーブルのアドレスを示します．（＄８ＡＤＡ）

＄０１Ｆ４　　　　　基本の関数の数を示します．（＄２Ｂ）

＄０１Ｆ５／＄０１Ｆ６　基本の関数の予約語テーブルのアドレスを示します．（＄８Ａ２Ｃ）

＄０１Ｆ７／＄０１Ｆ８　基本の関数のジャンプテーブルのアドレスを示します．（＄８８１６）

＄０１Ｆ９　　　　　ＤＩＳＫ用ステートメントの数を示します．（＄０６）

＄０１ＦＡ／＄０１ＦＢ　ＤＩＳＫ用ステートメントの予約語テーブルのアドレスを示します．（＄７３１４）

＄０１ＦＣ／＄０１ＦＤ　ＤＩＳＫ用ステートメントの処理アドレスを示します．（＄７３６Ｆ）

＄０１ＦＥ　　　　　ＤＩＳＫ用関数の数を示します．（＄０９）

＄０１ＦＦ／＄０２００　ＤＩＳＫ用関数の予約語テーブルのアドレスを示します．（＄７３３Ｃ）

＄０２０１／＄０２０２　ＤＩＳＫ用関数の処理アドレスを示します．（＄７３８６）

＄０２０３　　　　　拡張ステートメント（ＰＬＡＹなど）の数を示します．（＄０６）

＄０２０４／＄０２０５　拡張ステートメントの予約語テーブルのアドレスを示します．（＄８０３０）

＄０２０６／＄０２０７　拡張ステートメントの処理アドレスを示します．（＄８０５Ａ）

＄０２０８	拡張関数（ＬＰＯＳ）の数を示します．（＄０１）
＄０２０９／＄０２０Ａ	拡張関数の予約語テーブルのアドレスを示します．（＄８０７１）
＄０２０Ｂ／＄０２０Ｃ	拡張関数の処理アドレスを示します．（＄８０７７）
＄０２０Ｄ	再拡張ステートメント（未使用）の数を示します．（＄００）
＄０２０Ｅ／＄０２０Ｆ	再拡張ステートメントの予約語テーブルのアドレスを示します．（未使用なので＄００００）
＄０２１０／＄０２１１	再拡張ステートメントの処理アドレスを示します(未使用のため Syntax Error のアドレス（＄９２Ａ０）となっています)．
＄０２１２	再拡張関数（未使用）の数を示します．（＄００）
＄０２１３／＄０２１４	再拡張関数の予約語テーブルのアドレスを示します．(未使用なので＄００００)
＄０２１５／＄０２１６	再拡張関数の処理アドレスを示します．(未使用のため Syntax Error のアドレス（＄９２Ａ０）となっています．)
＄０２１７／＄０２１８	ＥＸＥＣのアドレスを示します．未設定の場合には Illegal Function Call のアドレス(＄９６６３)となります．
＄０２１９～＄０２２Ｃ	ＵＳＲ０～ＵＳＲ９のアドレスを示します．未設定の時はＥＸＥＣの場合と同様となります．

＄０２２D～＄０２A１	拡張のためのフックがあります．
＄０２６３～＄０２６５	ステートメントを実行するたびにＪＳＲしてきます．通常はＲＴＳとなっています．
＄０２６９～＄０２６Ｂ	ＭＯＮコマンドを実行する場合にＪＳＲしてきます．通常はＲＴＳとなっています．
＄０２８４～＄０２８６	ＴＥＲＭコマンドを実行する場合にＪＳＲしてきます．通常はＲＴＳとなっています．
＄０２９Ｃ～＄０２ＣＥ	ＥＸＴＩＲＱがかかったときに，ＪＳＲしてきます．通常はＲＴＳとなっています．
＄０２Ａ２～＄０２ＡＢ	ＴＡＢキーのカーソル停止位置を示します．
＄０２Ｂ２／＄０２Ｂ３	デバイスの登録テーブルのアドレスを示します．（＄０６ＥＣ）
＄０２Ｂ４／＄０２Ｂ５	論理機番と物理機番の対応テーブルのアドレスを示します．（＄０７０Ｃ）
＄０２Ｂ６～＄０２ＢＦ	ＯＮ～ＧＯＳＵＢの割り込み用のテーブルがあるアドレスを示します．

＄０２Ｂ６／＄０２Ｂ７	ＰＥＮ（現在未使用）	（＄０７１Ｅ）
＄０２Ｂ８／＄０２Ｂ９	ＣＯＭ（ｎ）	（＄０７２３）
＄０２ＢＡ／＄０２ＢＢ	ＫＥＹ（ｎ）	（＄０７３Ｃ）
＄０２ＢＣ／＄０２ＢＤ	ＴＩＭＥ	（＄０７６Ｅ）
＄０２ＢＥ／＄０２ＢＦ	ＩＮＴＥＲＶＡＬ	（＄０７７３）

＄０２Ｃ０～＄０２Ｃ９	ＣＯＭｎの登録テーブル（ＣＯＭＯ～ＣＯＭ４）です．つながっている場合には各ルーチンベクトルのあるアドレスを示します．
＄０２Ｄ５	ＣＡＳ０においてのエラーの種類を示します． ０：エラーなし １：チェックサムエラー ２：読み込みエラー（ＢＩＯＳで生じるもの）
＄０２ＤＢ	ファイルのタイプを示します． ０：ＢＡＳＩＣのソース １：ＢＡＳＩＣのデータ ２：マシン語ファイル
＄０２ＤＣ	ファイルのアスキーフラグを示します． ＄００：バイナリファイル ＄ＦＦ：アスキーファイル
＄０２ＤＤ	ファイル名の長さを示します．
＄０２ＤＥ～＄０２Ｅ５	ファイル名が入ります．８文字に満たない部分には＄２０（空白）を埋めます．
＄０２Ｅ６	入出力を行なうデバイスの物理機番が入ります．
＄０２Ｅ７～＄０２ＥＣ	ファイルオプションが入ります．
＄０２Ｆ４～＄０２Ｆ９	直前にタイマを参照した時の時刻です．
＄０２ＦＡ～＄０２ＦＦ	日付が格納されています．＄０２ＦＡ，Ｂが年，＄２ＦＣ，Ｄが月，＄０２ＦＥ，Ｆが日を表わしています．

＄０３０Ｄ	画面の行数（ＷＩＤＴＨ文第２パラメータ）を示します．
＄０３０Ｅ	スクロール開始行を示します．
＄０３０Ｆ	スクロール終了行を示します．
＄０３１０	ファンクションキー表示フラグです． ０：非表示　　０以外：表示
＄０３１３	この値が０でない場合，ＢＲＥＡＫキーが押されたことを示します．
＄０３１４／＄０３１５	ＭＯＮコマンドでのリードライトを行なうアドレスを示します．
＄０３１８～＄０３１Ｅ	ＲＮＤのためのワークエリアです．
＄０３１Ｆ～＄０３３８	型宣言を行なった変数の型を示しています（Ａ～Ｚの２６バイト）．宣言されていない場合には＄０４(単精度）となっています．
＄０３３Ａ～＄０４３Ｃ	キー入力されたＢＡＳＩＣテキストの中間言語に変換されたものが格納されます．
＄０４３Ｄ～＄０５３Ｃ	１行入力のときに入力された文字列が格納されます．
＄０５９Ａ	タイマなどの割り込みがおきた回数を示します．
＄０５９Ｄ／＄０５９Ｅ	ユーザーが使用できるＲＡＭの最終アドレス－１の値を示します．
＄０５Ａ０～＄０５Ａ７	ＢＡＳＩＣ内部からＢＩＯＳを使用するときに，ＲＣＢをセットする領域です．

＄０５Ａ８	画面のコンソールコントロールに用いる値です．bit 0がカーソルの表示を示します．
＄０５ＡＡ／＄０５ＡＢ	ＬＰＴ０のルーチンベクトルのあるアドレスを示しています．（＄０７７８）
＄０５ＡＣ	この値を０以外にしておくと，論理機番０による画面出力の際，プリンタにも出力されます．
＄０５ＡＤ	単色表示であるかを示すフラグです．０以外のとき単色表示であることを示します．
＄０５ＡＦ／＄０５Ｂ０	ＰＦキーが割り込みに設定されている場合に該当するビットが１となります．＄０５Ｂ０のbit 0がＰＦ１に対応します．
＄０５Ｂ１／＄０５Ｂ２	この値が０でない場合，ＰＦキー割り込みがあるとこの値をアドレスとしてジャンプします．ジャンプしたルーチンの最後はＲＴＩで終了しなければなりません．
＄０５ＢＦ	プリンタに対する改行制御をこの値によって行ないます．くわしくは，「ユーザーズマニュアルシステム解説」を御参照下さい． ０：Ｆ指定 １：Ｎ指定
＄０５Ｄ２	＄ＦＤ０２に書き込んだ値を保持しています．通常は＄４０（Rx ＲＤＹだけ使用）となっています．
＄０５ＤＤ	ＳＣＲＥＥＮ文の第１パラメータを保持しています．
＄０５ＤＥ	ＳＣＲＥＥＮ文の第２パラメータを保持しています．

＄０５ＤＦ～＄０５Ｅ１　メインＣＰＵへの２．０３ｍｓごとのＴＩＭＥＲ割り込みのときにＪＳＲしてきます．ＰＬＡＹ文でこの割り込みを使用しているので，通常はＪＭＰ＄ＥＤ７２となっています．

＄０５Ｅ２～＄０５Ｅ４　メインＣＰＵへのＫＥＹ割り込みのときＪＳＲしてきます．通常はＲＴＳとなっています．

＄０５ＥＤ～＄０６ＥＢ　ＣＡＳ０の入出力バッファです．ＰＬＡＹ文でも使用しています．

＄０６ＥＣ～＄０７０Ｂ　ディスクを除く，入出力デバイスの登録テーブルです．＄０６ＥＣ，Ｄに物理機番０に対応したデバイスのルーチンベクトルがあるアドレスが書かれています．

この値が０の物理機番は使用されません．

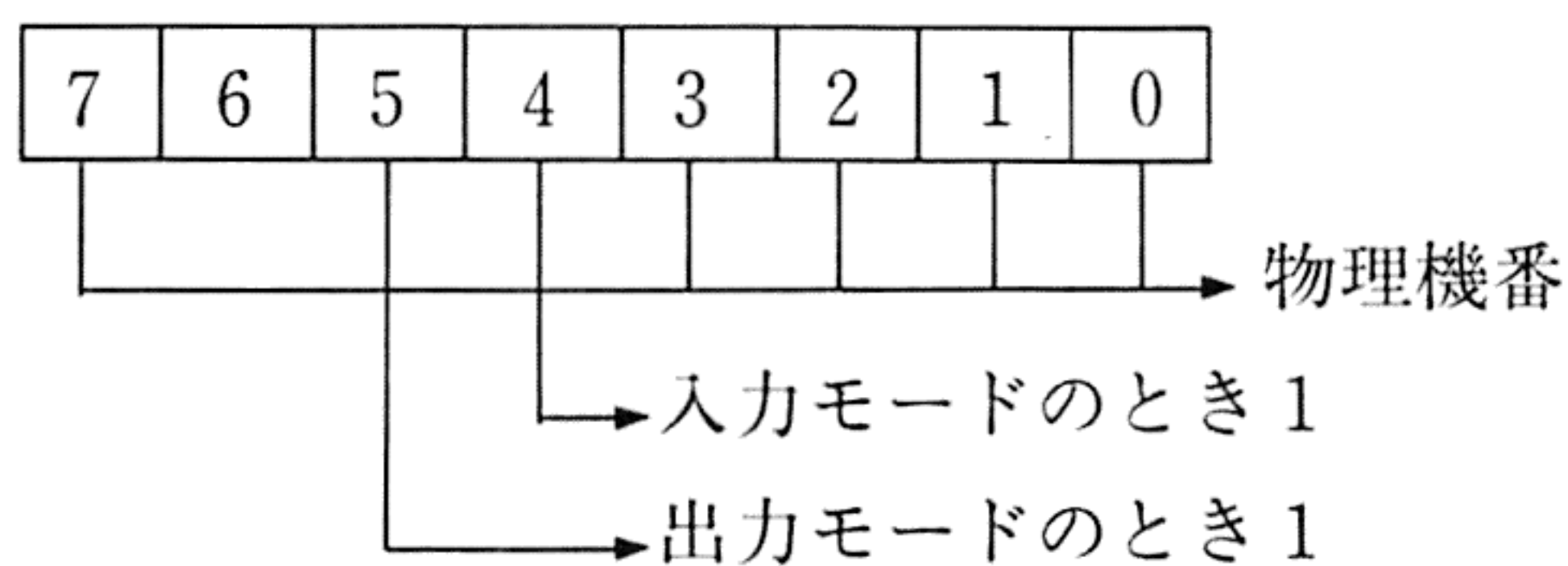
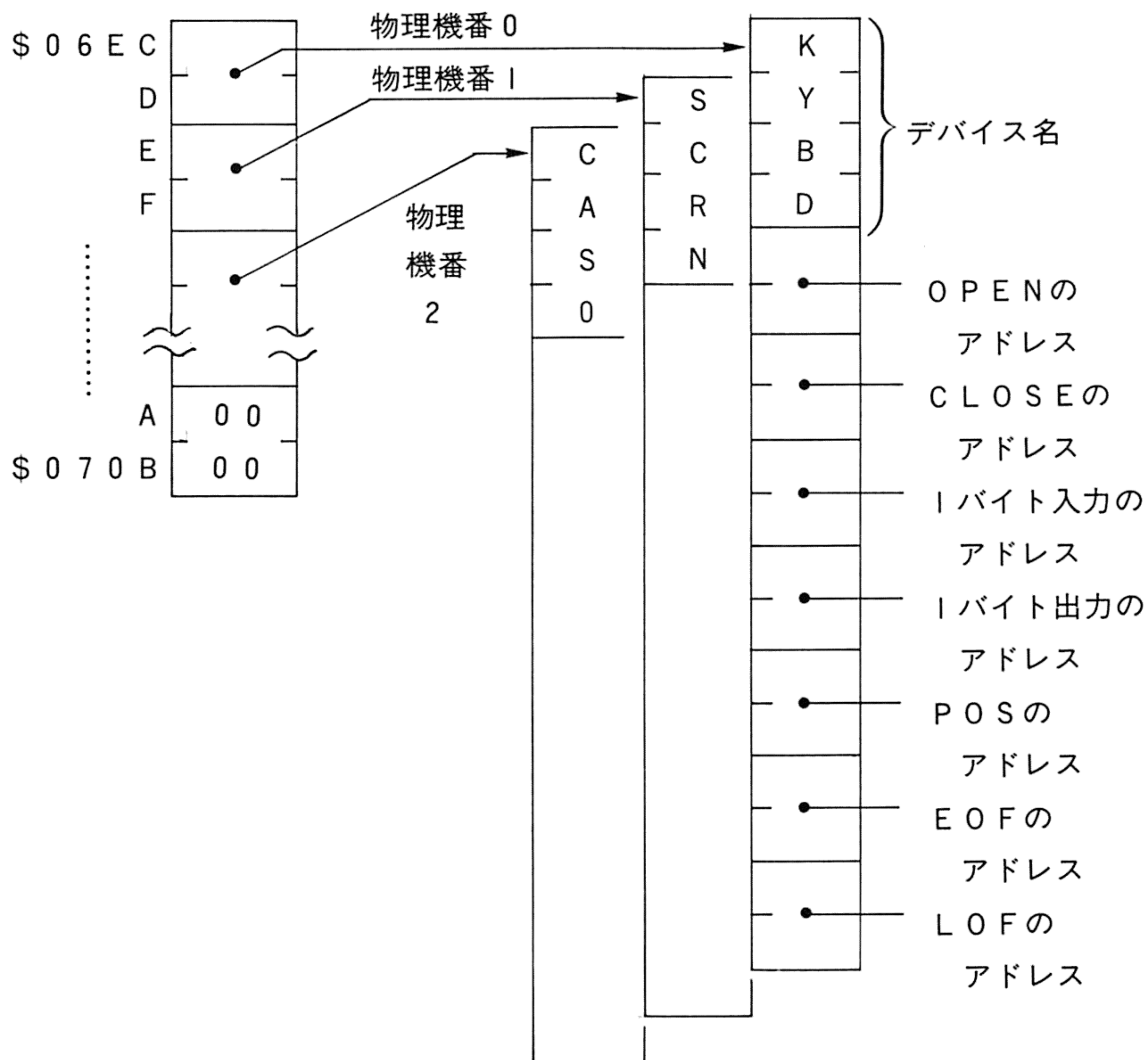
＄０７０Ｃ～＄０７１Ｄ　論理機番と物理機番の対応テーブルです．論理機番０～１７までの１８バイトです．

7	6	5	4	3	2	1	0

7, 3, 2, 1, 0 → 物理機番
4 → 入力モードのとき１
5 → 出力モードのとき１

登録テーブル　　　　ルーチンベクトル

$06EC　物理機番 0 → KYBD
D
E　物理機番 1 → SCRN
F　物理機番 2 → CAS0
A　00
$070B　00

デバイス名
OPENのアドレス
CLOSEのアドレス
1バイト入力のアドレス
1バイト出力のアドレス
POSのアドレス
EOFのアドレス
LOFのアドレス

	KYBD:	SCRN:	CAS0:	LPT0:	COM0:
OPEN	$DB1F	$D99E	$CA06	$D9A5	$D1AD
CLOSE	$DB25	$D9A4	$CAB6	$D615	$D198
IN	$DB54	$DB54	$CB02	$CEF4	$D2B5
OUT	$D9D9	$D9D9	$CB1E	$D617	$D25C
POS	$D992	$D992	$CB4D	$D39C	$D39C
EOF	$CEF4	$CEF4	$C9FA	$CEF4	$D3A5
LOF	$CEF4	$CEF4	$CA02	$CEF4	$D3B4

＄０７１Ｅ～＄０７７７　　ＯＮ～ＧＯＳＵＢの割り込み用のテーブルです．

＄０７１Ｅ～＄０７２２　ＰＥＮ（現在未使用）
＄０７１３～＄０７３Ｂ　ＣＯＭ（ｎ）
＄０７３Ｃ～＄０７６Ｄ　ＫＥＹ（ｎ）
＄０７６Ｅ～＄０７７２　ＴＩＭＥＲ
＄０７７３～＄０７７７　ＩＮＴＥＲＶＡＬ

各割り込みについて５バイトずつ使用されていて，意味は以下の 様になります．

Byte　0：　０以外のときＯＮ状態であることを示します．
Byte　1：　０以外のときＳＴＯＰ状態であるか，割り込み処理ルーチンを実行中であることを示します．
Byte　2：　０以外のとき割り込みがあったことを示します．
Byte 3,4：　ＯＮ～ＧＯＳＵＢで指定された行のあるアドレスを示します．

＄０７７８～＄０７８Ｃ　　ＬＰＴ０のルーチンベクトルがあります．

＄０７９０　　ＲＯＭモードの場合このアドレス以降にＢＡＳＩＣ ＴＥＸＴが格納されます．

7. Circuit Diagram

本章では，ＦＭ－７の全回路図を紹介します．

これらの回路図は全て秀和システムトレーディング株式会社において調査したもので，メーカーが保障したものではありません．あくまでも参考図としてお使いください．したがいまして，本回路図集に関してメーカー等に直接問い合わせることはご遠慮ください．

本回路図集は，ＦＭ－７本体をブロック図，回路図，実装図に分類しまとめたものです．

Ｉ．ブロック図

II．回　路　図

1．メインＣＰＵ
2．クロック制御
3．メインＩ／Ｏアドレス
4．メインＲＯＭ
5．メインＲＡＭ
6．メイン・ブザー
7．メインプリンタ・カセット
8．共有メモリ
9．サブＣＰＵ
10．サブ・アドレス・デコード
11．サブＲＯＭ／ＲＡＭ
11．サブＲＥＧ／ＦＬＡＧ
13．サブＣＲＴ／ＣＮＴＲＬ
14．サブＣＲＴアドレス
15．サブＣＲＴ ＲＡＭ
16．サブＣＲＴインタフェース
17．キーインタフェース
18．拡張バス・バッファ
19．コネクタ拡張・Ｚ８０
20．コネクタ（ＰＳＧ，漢字ＲＯＭ／ＲＳ－２３２Ｃ）
21．メインＰＳＧ
22．電源
23．漢字ＲＯＭ
24．コネクタ漢字，電源
25．キー・マトリックス

III．部品実装図

1．本体部品番号図
2．本体部品配置図
3．漢字ＲＯＭ，ＰＳＧ基板部品番号図，配置図

1．メインCPU

M154
MBM2764-20(SC5)

M153
MBM2732A-20(SC4)

M141
MB8128

M123
MB8128

J1

J2

CB54 0.1μ

CB53 0.1μ

CB44 0.1μ

CB35 0.1μ

SDATABUS

SADDRBUS

・SROMD

・SROMSEL

・SRAM1CS

・SRAM2CS

・SWTOE

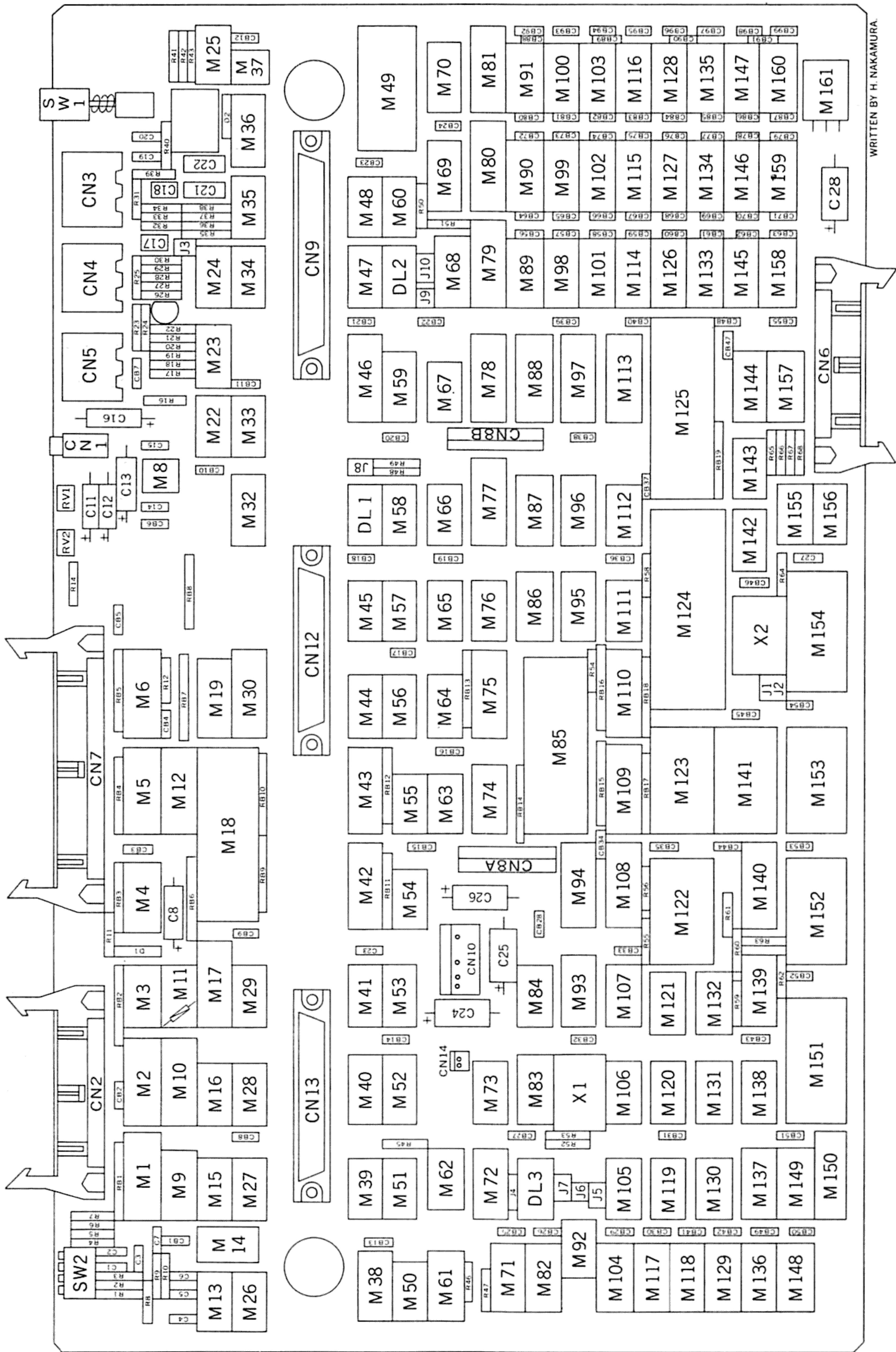
・SRDOE

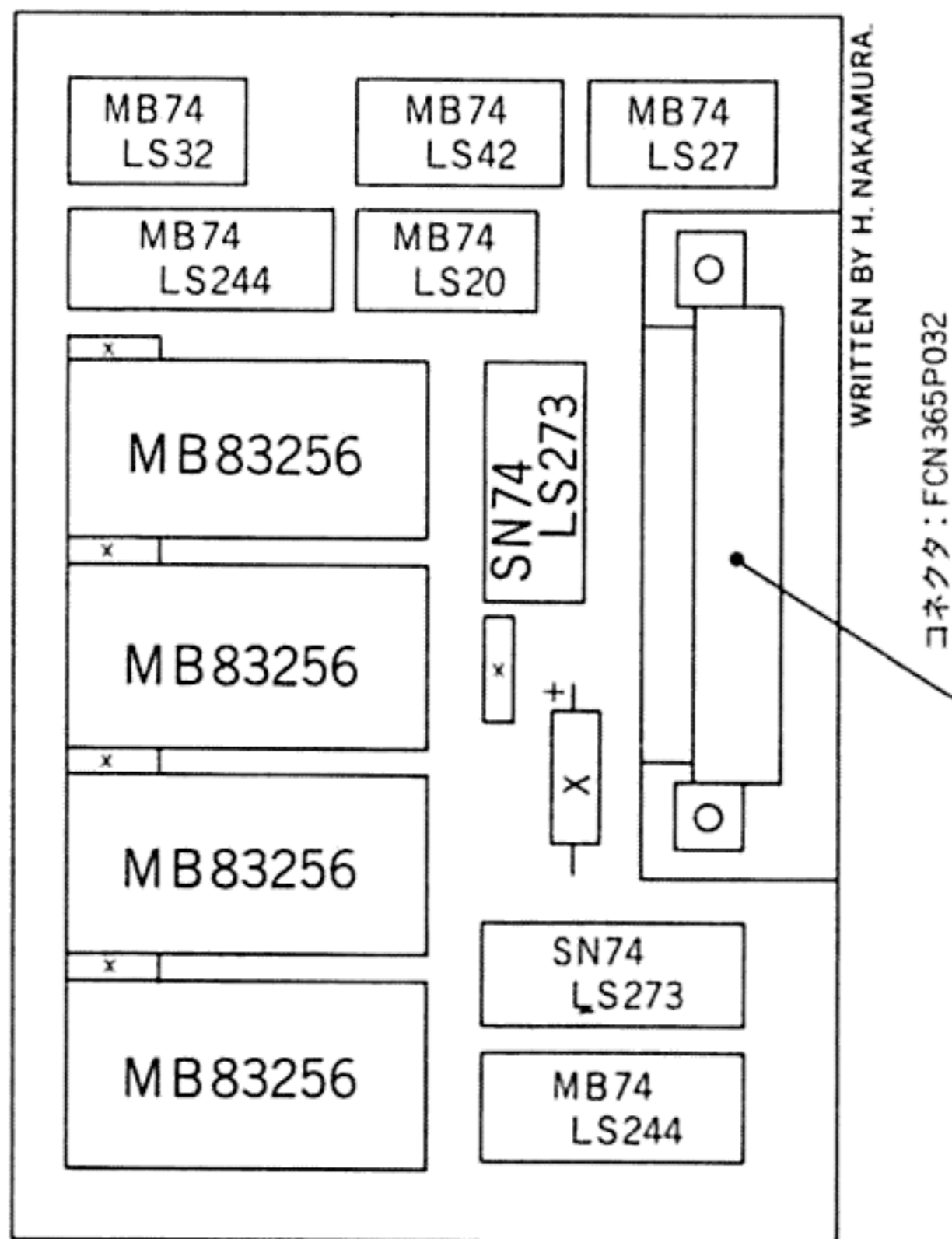
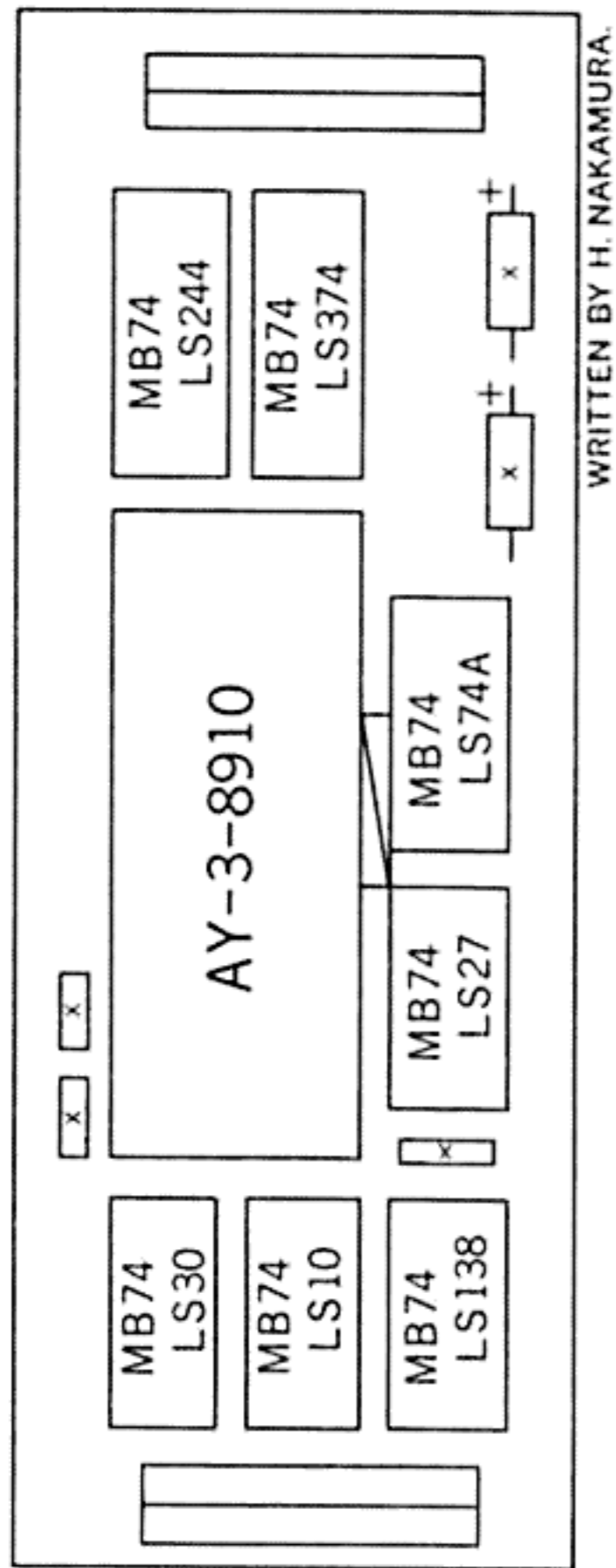
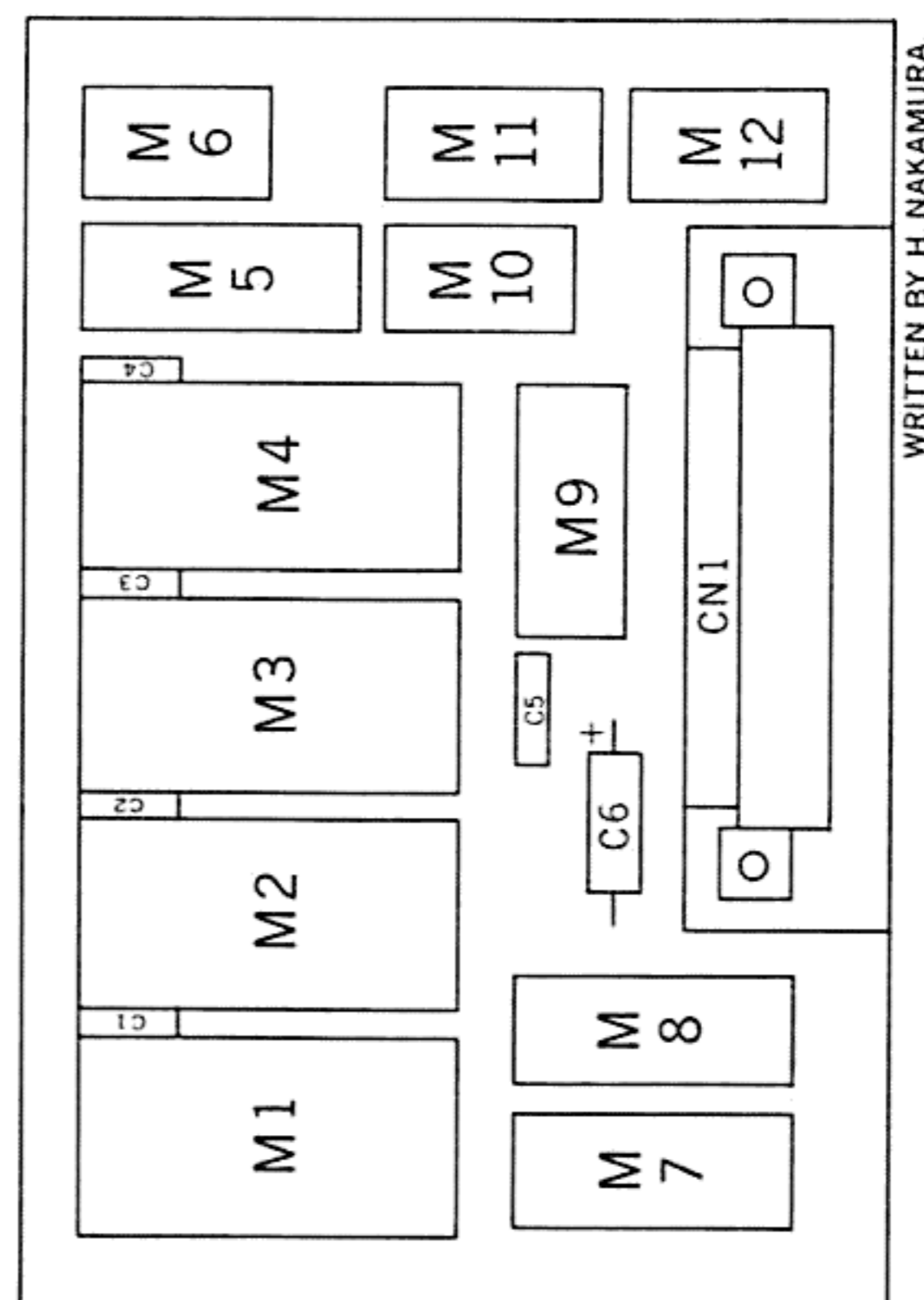
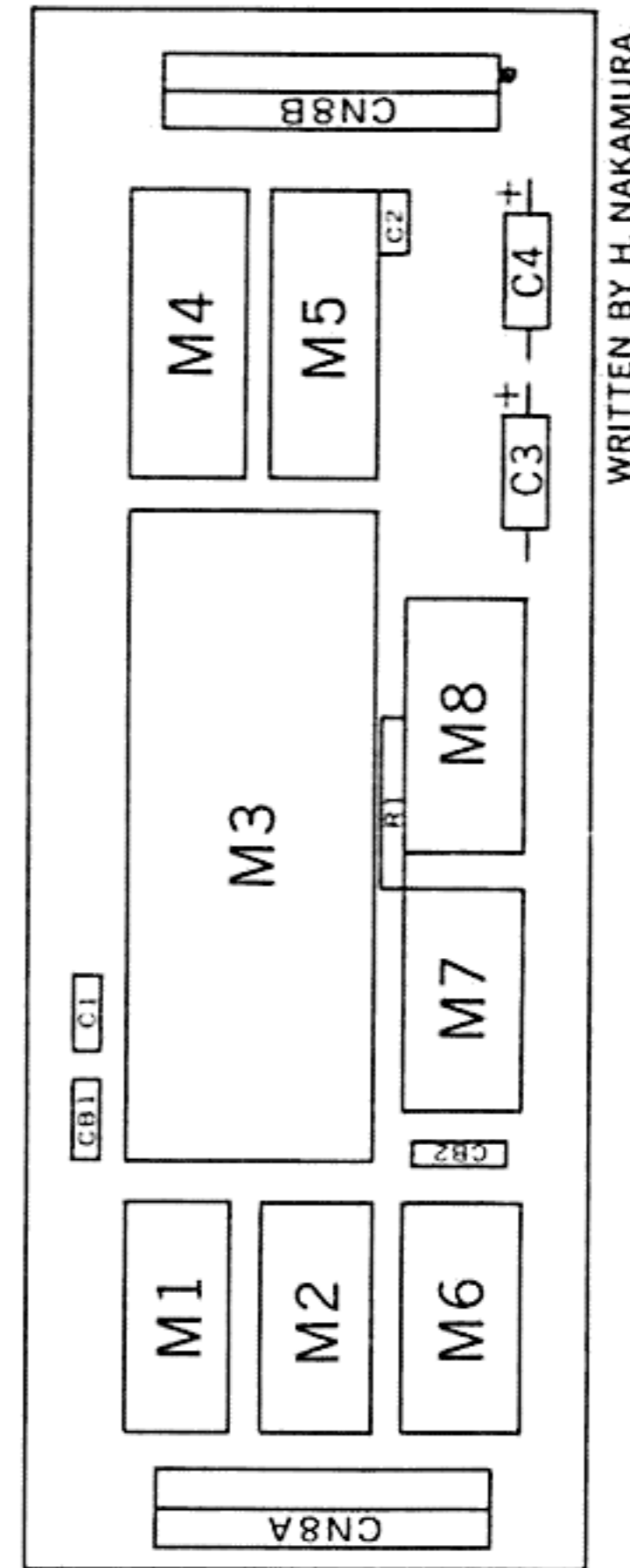
SVCASB
SVCASR
SVCASG
M48
MB74LS32
M47
MB74LS08
SDRAMV1
SDRAMV2
SDRAMV3
SCASSEL
SADRSEL
R51
620
R50
620
+5
DL2
J10
J9
SVWE
SVRAS
M59
MB462
SCPUWE
SCPURAS
SCPUCAS
SVDRAS
SVDCAS
SBLANK

21. メインPSG

	K0 (17)	K1 (18)	K2 (19)	K3 (20)	K4 (21)	K5 (22)	K6 (23)	K7 (24)
Ka (1)	ESC S1B	1 S31	Q S6B	A S61	Z S7A	TAB S09		
Kb (2)	PF 1 S8001	2 S32	W S77	S S73	X S78			
Kc (3)	PF2 S8002	3 S33	E S65	D S64	C S63			
Kd (4)	PF3 S8003	4 S34	B S62	F S66	V S76			
Ke (5)	PF4 S8004	5 S35	T S74	G S67	B S62	S P S20		
Kf (6)	PF5 S8005	6 S36	Y S79	H S68	N S6E	= S3D	− S2D	
Kg (7)	PF6 S8006	7 S37	U S75	J S6A	M S6D	9 S39	+ S2B	
Kh (8)	PF7 S8007	8 S38	I S69	K S6B	, S2C	8 S38	／ S2F	
Ki (9)	PF8 S8008	9 S39	O S6F	L S6C	. S2E	7 S37	＊ S27	
Kj (10)	PF9 S8009	0 S30	P S70	; S3B	／ S2F	4 S34	1 S31	
Kk (11)	PF10 $8010	- S2D	@ S40	: S3A	▼▼ S22	5 S35	2 S32	
Kl (12)		^ S5E	[S5B	] S5B		6 S36	3 S33	
Km (13)		¥ S5C	⇦ S08	S0D ↵		, S2C	S0D ↵	
Kn (14)		INS S12	E L S05	DEL S7F	↑ S1E	← S1D	0	
Ko (15)		CLS S0C	DUP S11	HOME S0B	↓ S1F	→ S1E	.	
Kp (16)			CAP	CTRL	SHIFT	GRAPH	カナ	

[- - -]で囲まれた部分は"テンキー"です

FM-7 本体 メイン基板 実装図(A)
—部品番号図—

WRITTEN BY H. NAKAMURA.

FM-7 本体 メイン基板 実装図(B)
—部品配置図—

8. Appendix

A：BIOS処理アドレス

リクエスト番号		ラベル名	処理アドレス
No. 0	$00	ANALGP	$F1E1
No. 1	$01	MOTOR	$F2A3
No. 2	$02	CTBWRT	$F2B0
No. 3	$03	CTBRED	$F330
No. 4	$04	INTBBL	$F4C8
No. 5	$05	SCREEN	$F998
No. 6	$06	WRTBBL	$F529
No. 7	$07	REDBBL	$F529
No. 8	$08	RESTOR	$FE02
No. 9	$09	DWRITE	$FE05
No. 10	$0A	DREAD	$FE08
No. 11	$0B	-------	$F5A8
No. 12	$0C	BEEPON	$F5A1
No. 13	$0D	BEEPOF	$F5A4
No. 14	$0E	LPOUT	$F5AA
No. 15	$0F	HDCOPY	$F7B8
No. 16	$10	SUBOUT	$F5F1
No. 17	$11	SUBIN	$F5F1
No. 18	$12	INPUT	$F655
No. 19	$13	INPUTC	$F713
No. 20	$14	OUTPUT	$F72D
No. 21	$15	KEYIN	$F78F
No. 22	$16	KANJIR	$F21F
No. 23	$17	LPCHK	$F5C7
No. 24	$18	BIINIT	$FB05
No. 25	$19	-------	$F5A8
No. 26	$1A	-------	$F5A8
No. 27	$1B	-------	$F5A8

B：中間言語，処理アドレス

中間コード		処理アドレス
$80	END	$8FD0
$81	FOR	$A203
$82	NEXT	$A2C7
$83	DATA	$90A0
$84	DIM	$94C5
$85	READ	$BFF3
$86	LET	$9195
$87	GO	$902D
$88	RUN	$9012
$89	IF	$90F1
$8A	RESTORE	$8FBC
$8B	RETURN	$9071
$8C	REM	$90A3
$8D	'	$90A3
$8E	STOP	$8FD9
$8F	ELSE	$90A3
$90	TRON	$A01A
$91	TROFF	$A01B
$92	SWAP	$A784
$93	DEFSTR	$9F73
$94	DEFINT	$9F76
$95	DEFSNG	$9F79
$96	DEFDBL	$9F7C
$97	ON	$01EC
$98	HARDC	$ADDC
$99	RENUM	$A9CA
$9A	EDIT	$D8AC
$9B	ERROR	$A739
$9C	RESUME	$A744
$9D	AUTO	$9FE7
$9E	DELETE	$9FBC

```
$9F  TERM       $D52D
$A0  WIDTH      $C87B
$A1  UNLIST     $C912
$A2  MON        $ABF4
$A3  LOCATE     $C8F0
$A4  CLS        $DA80
$A5  CONSOLE    $C814
$A6  PSET       $DFA6
$A7  PRESET     $DFAC
$A8  MOTOR      $D758
$A9  SKIPF      $CAE6
$AA  SAVE       $CD3E
$AB  LOAD       $CF31
$AC  MERGE      $CF29
$AD  EXEC       $C76F
$AE  OPEN       $CEA8
$AF  CLOSE      $CE73
$B0  FILES      $CBB5
$B1  COM        $D444
$B2  KEY        $DB94
$B3  PAINT      $E4C8
$B4  BEEP       $DB2A
$B5  COLOR      $C7CA
$B6  LINE       $E064
$B7  DEF        $A7D9
$B8  POKE       $9A30
$B9  PRINT      $9AB3
$BA  CONT       $9002
$BB  LIST       $CD01
$BC  CLEAR      $C77B
$BD  RANDOMIZE  $BBE5
$BE  WHILE      $AD54
$BF  WEND       $AD81
$C0  NEW        $8F32
```

```
$C1  GET          $E7EF
$C2  PUT          $E80A
$C3  CIRCLE       $E552
$C4  CONNECT      $DCFA
$C5  SYMBOL       $DDA8
$C6  GCURSOR      $DE33
$C7  BUBINI       $EB8E
$C8  BUBW         $EB91
$C9  BUBR         $EB94
$CA  KILL         $EB9A
$CB  INTERVAL     $E3BA
$CC  TAB(
$CD  TO
$CE  SUB
$CF  FN
$D0  SPC(
$D1  USING
$D2  USR
$D3  ERL
$D4  ERR
$D5  OFF
$D6  THEN
$D7  NOT
$D8  STEP
$D9  +
$DA  -
$DB  *
$DC  /
$DD  ^
$DE  AND
$DF  OR
$E0  XOR
$E1  EQV
$E2  IMP
```

```
    $E3 MOD
    $E4 ¥
    $E5 >
    $E6 =
    $E7 <

    $E8 DSKINI        $73A1
    $E9 DSKO$         $7565
    $EA NAME          $7CE6
    $EB FIELD         $7D4E
    $EC LSET          $7DA8
    $ED RSET          $7DA7

    $EE CHAIN         $808E
    $EF ERASE         $8448
    $F0 LLIST         $CD88
    $F1 LPRINT        $9A7F
    $F2 SOUND         $EBBA
    $F3 PLAY          $EC72

$FF $80 SGN           $B3BE
$FF $81 INT           $B472
$FF $82 ABS           $B3D5
$FF $83 FRE           $972C
$FF $84 POS           $C92A
$FF $85 SQR           $BAEE
$FF $86 LOG           $B0AA
$FF $87 EXP           $BB5B
$FF $88 COS           $BE60
$FF $89 SIN           $BE66
$FF $8A TAN           $BEB1
```

```
$FF  $8B  ATN       $BEF5
$FF  $8C  PEEK      $9A26
$FF  $8D  LEN       $992A
$FF  $8E  STR$      $9752
$FF  $8F  VAL       $99CA
$FF  $90  ASC       $9947
$FF  $91  CHR$      $9933
$FF  $92  CINT      $BC74
$FF  $93  CSNG      $BCE0
$FF  $94  CDBL      $BCCD
$FF  $95  FIX       $B461
$FF  $96  SPACE$    $9D7B
$FF  $97  HEX$      $9E2C
$FF  $98  OCT$      $9E2D
$FF  $99  LOF       $D11D
$FF  $9A  EOF       $D11A
$FF  $9B  PEN       $025A  $0257
$FF  $9C  LEFT$     $9952
$FF  $9D  RIGHT$    $996F
$FF  $9E  MID$      $9979  $9DC1
$FF  $9F  INSTR     $9CD4
$FF  $A0  SCREEN    $EAF6  $EB4B
$FF  $A1  ANPORT    $AED2
$FF  $A2  VARPTR    $A952
$FF  $A3  STRING$   $9D59
$FF  $A4  RND       $BBFE
$FF  $A5  INKEY$    $DB84
$FF  $A6  INPUT     $D15C  $BFDD
$FF  $A7  CSRLIN    $C923
$FF  $A8  POINT     $DF78
$FF  $A9  TIME      $E11A  $E1E4
$FF  $AA  DATE      $E190  $E291
```

ステートメントの場合

```
$FF  $AB  DSKF          $7C15
$FF  $AC  CVI           $7C3D
$FF  $AD  CVS           $7C40
$FF  $AE  CVD           $7C43
$FF  $AF  MKI$          $7C55
$FF  $B0  MKS$          $7C58
$FF  $B1  MKD$          $7C5B
$FF  $B2  LOC           $7C72
$FF  $B3  DSKI$         $75A4


$FF  $B4  LPOS          $C947
```

◀テーブル・アドレス一覧▶

中間コード	予約語テーブル	ジャンプ・テーブル	
$80~$CB	$8890~$89E6	$8ADA~$8B71	
$CC~$E7	$89E7~$8A2B		
$E8~$ED	$7314~$732F	$7330~$733B	*
$EE~$F3	$8030~$804D	$804E~$8059	
$FF80~ $FFAA	$8A2C~$8AD9	$8816~$886B	
$FFAB~ $FFB3	$733C~$735C	$735D~$736E	*
$FFB4	$8071~$8074	$8075~$8076	

*：DISK モード

C：マシン語モニタ処理アドレス

コマンド		処理アドレス
G	(指定アドレスへ分岐)	＄ＡＣ２Ｆ
D	(指定アドレスから６４バイト表示)	＄ＡＣＣ４
R	(レジスタ表示・変更)	＄ＡＣＦＢ
M	(メモリ内容変更)	＄ＡＣ９５

D：キャラクタコード表

上位４ビット→

下位４ビット↓

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		D_E		0	@	P	`	p				ー	タ	ミ		
1	S_H	D_1	!	1	A	Q	a	q			。	ア	チ	ム		円
2	S_X	D_2	"	2	B	R	b	r			「	イ	ツ	メ		年
3	E_X	D_3	#	3	C	S	c	s			」	ウ	テ	モ		月
4	E_T	D_4	$	4	D	T	d	t			、	エ	ト	ヤ		日
5	E_Q	N_K	%	5	E	U	e	u			・	オ	ナ	ユ		時
6	A_K	S_N	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		分
7	B_L	E_B	'	7	G	W	g	w			ァ	キ	ヌ	ラ		秒
8	B_S	C_N	(	8	H	X	h	x			ィ	ク	ネ	リ	♠	〒
9	H_T	E_M	)	9	I	Y	i	y			ゥ	ケ	ノ	ル	♥	市
A	L_F	S_B	*	:	J	Z	j	z			ェ	コ	ハ	レ	♦	区
B	H_M	E_C	+	;	K	[	k	{			ォ	サ	ヒ	ロ	♣	町
C	C_L	→	,	<	L	¥	l	¦			ャ	シ	フ	ワ	●	村
D	C_R	←	−	=	M	]	m	}			ュ	ス	ヘ	ン	○	
E	S_O	↑	.	>	N	∧	n	~			ョ	セ	ホ	゛		
F	S_I	↓	/	?	O	_	o	D_L			ッ	ソ	マ	゜		

E：メモリ・マップ

◀メインCPU▶

$0000
$4000
RAM
$8000
BASIC ROM
$C000
$FC00
$FFFF
RAMモード
ROMモード

$FC00
RAM (128B)
$FC80
共有RAM (128B)
サブCPU
$FD00
I/O 領域
$FE00
ブートROM
$FF00
$FFE0
RAM
$FFF0
$FFFF

$FFF0	RAM	
$FFF2		SWI3ベクトル
$FFF4		SWI2ベクトル
$FFF6		FIRQベクトル
$FFF8		IRQベクトル
$FFFA		SWIベクトル
$FFFC		NMIベクトル
$FFFE	ROM	RESETベクトル

◀サブCPU▶

アドレス	内容
$0000	VRAM (B)
$4000	VRAM (R)
$8000	VRAM (G)
$C000	コンソールバッファ
$D000	(斜線部)
$E000	キャラクタROM
	ROM (CRTモニタ)
$FFFF	

アドレス	内容
$D000	ワークRAM (896B)
$D380	共有RAM (メインCPU)
$D400	I/O領域
$D7FF	

F：マシン語プログラム入力のしかた

本書では，サンプルプログラムをアセンブル・リストの形で載せています．このため，アセンブラを持たない読者の方は，アセンブルリストの左側に出力されている１６進コードを，モニタによって入力しなければなりません．この際，注意すべき点に触れながら，入力方法を示します．

まず，アセンブル・リストの出力形式を見て下さい．このリストは，富士通が提供している「アブソリュート・アセンブラ（FM７用)」によって出力された物です．

① 行番号
② アドレス
③ マシン語命令コード（２バイトのときもある)
④ マシン語オペランド
⑤ 相対分岐命令（BSR，BEQなど）の分岐先のアドレス

このうち，入力しなければならないのは，③，④の部分です．

具体的には，Mコマンドで，５０００番地から，ワクのかかっている部分を入力します．このとき，一行入力し終わるごとにアドレスの値を確認すると誤りが少なくなります．(特に，相対分岐命令の分岐先を入力しないように気をつけて下さい．)

```
PAGE  001 (          ,           )

01000                             *
01010                             *          ﾆｭｳﾘｮｸ ｾﾂﾒｲﾖｳ ｻﾝﾌﾟﾙ ﾌﾟﾛｸﾞﾗﾑ
01020                             *
01030                                    OPT     M,NOO,NOG,NOS,PAGE=255
01040             9BDB            MESSAG EQU     $9BDB    output message from (X+1) till
01050             FD03            BEEP   EQU     $FD03    I/O address of BEEP
01060  5000                              ORG     $5000
01070             5000            START  EQU     *
01080                             *
01090  5000 30    8D 0013         NEXT   LEAX    MES-1,PCR
01100  5004 BD    9BDB                   JSR     MESSAG
01110  5007 86    41                     LDA     #%01000001
01120  5009 B7    FD03                   STA     BEEP
01130  500C 20    F2     5000            BRA     NEXT
01140                             *
01150  500E       000A            WORK   RMB     10
01160  5018       0D0A            MES    FDB     $0D0A
01170  501A       53                     FCC     'SAMPLE PROGRAM'
01180  5028       00                     FCB     0
01190             5000                   END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5028
PROGRAM ENTRY ADDR=5000
```

次に疑似命令（本書では，ＦＣＣ，ＦＣＢ，ＦＤＢ，ＥＱＵ，ＯＲＧ，ＯＰＴ，ＲＭＢ，ＥＮＤを使用）のところの入力方法ですが，ＥＱＵ，ＥＮＤ命令の左に出力されているコードは入力しません．ＲＭＢの場合には，左に示されているバイト数だけアドレスを飛ばします（適当な値を入力します）．

ＦＤＢ，ＦＣＢの場合には，右に書かれている値を入力するようにします．右に複数の値がカンマ（，）で区切られて示されている場合には，それらすべてを順に入力します．この場合，左には最初の値しか出力されていないので注意して下さい．

例：　６０００　０１　　　ＦＣＢ　１，２，３，４

この場合には，６０００番地から順に１，２，３，４と６００３番地まで入力する．

ＦＣＢの場合には，さらに複雑になります．ＦＣＢの左には，右の文字列の最初の１バイトしか出力されていないので，２文字目以降のアスキーコードを入力していくことになります．

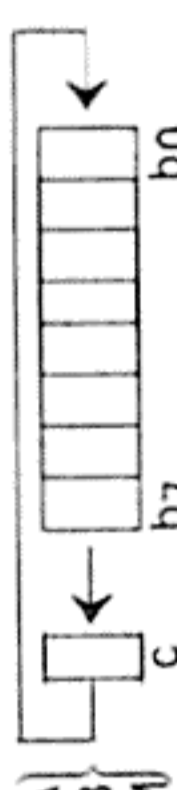
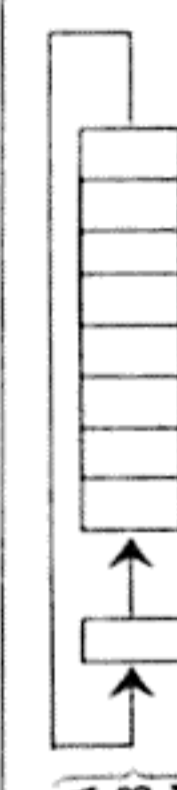
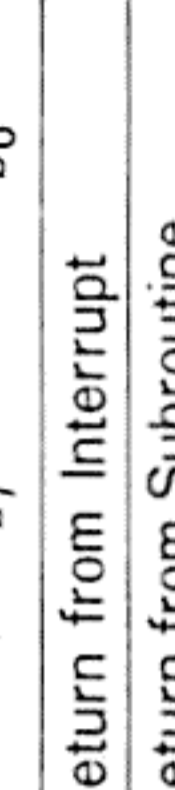
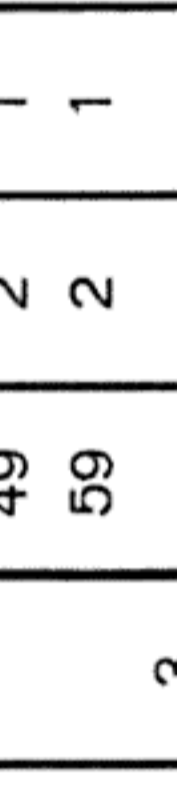
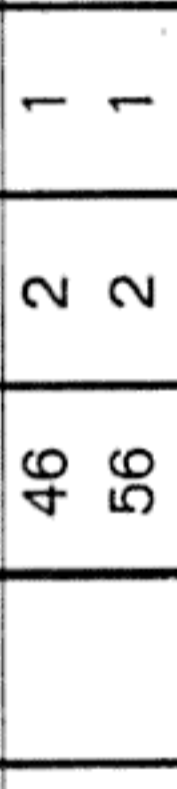
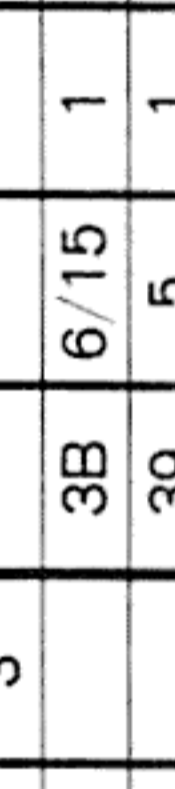
例：　６０００　４１　　　ＦＣＣ　'ＡＢＣ'

この場合には，６０００番地から順に＄４１，＄４２，＄４３と入力します．

以上のようにして入力したプログラムは，リスト末尾のアドレスにしたがってＳＡＶＥ，ＥＸＥＣして下さい．

```
MON

*M5000
5000 00-30
5001 00-8D
5002 00-00
5003 00-13
5004 00-BD
5005 00-9B
5006 00-DB
5007 00-86
5008 00-41
5009 00-B7
500A 00-FD
500B 00-03
500C 00-20
500D 00-F2
500E 00-00
500F 00-00
5010 00-00
5011 00-00
5012 00-00
5013 00-00
5014 00-00
5015 00-00
5016 00-00
5017 00-00
5018 00-0D
5019 00-0A
501A 00-53
501B 00-41
501C 00-4D
501D 00-50
501E 00-4C
501F 00-45
5020 00-20
5021 00-50
5022 00-52
5023 00-4F
5024 00-47
5025 00-52
5026 00-41
5027 00-4D
5028 00-00
5029 00-.

*
```

```
*D5000

5000 30 8D 00 13 BD 9B DB 86
5008 41 B7 FD 03 20 F2 00 00
5010 00 00 00 00 00 00 00 00
5018 0D 0A 53 41 4D 50 4C 45
5020 20 50 52 4F 47 52 41 4D
5028 00 00 00 00 00 00 00 00
5030 00 00 00 00 00 00 00 00
5038 00 00 00 00 00 00 00 00
*

Break
Ready
```

G．MC6809インストラクション・コード表

1982 © MOTOROLA INC.

インストラクション	ニーモニック	アドレッシングモード															動作	5	3	2	1	0
		イミデエイト			ダイレクト			インデックス[1]			エクステンド			インヘレント				H	N	Z	V	C
		Op	~	#	Op	~	#	Op	~	#	Op	~	#	Op	~	#						
ABX														3A	3	1	B+X→X(Unsigned)	●	●	●	●	●
ADC	ADCA	89	2	2	99	4	2	A9	4+	2+	B9	5	3				A+M+C→A	↕	↕	↕	↕	↕
	ADCB	C9	2	2	D9	4	2	E9	4+	2+	F9	5	3				B+M+C→B	↕	↕	↕	↕	↕
ADD	ADDA	8B	2	2	9B	4	2	AB	4+	2+	BB	5	3				A+M→A	↕	↕	↕	↕	↕
	ADDB	CB	2	2	DB	4	2	EB	4+	2+	FB	5	3				B+M→B	↕	↕	↕	↕	↕
	ADDD	C3	4	3	D3	6	2	E3	6+	2+	F3	7	3				D+M:M+1→D	●	↕	↕	↕	↕
AND	ANDA	84	2	2	94	4	2	A4	4+	2+	B4	5	3				A∧M→A	●	↕	↕	0	●
	ANDB	C4	2	2	D4	4	2	E4	4+	2+	F4	5	3				B∧M→B	●	↕	↕	0	●
	ANDCC	1C	3	2													CC∧IMM→CC					6
ASL	ASLA													48	2	1	A, B, M: C ← b7 … b0 ← 0	7	↕	↕	↕	↕
	ASLB													58	2	1		7	↕	↕	↕	↕
	ASL				08	6	2	68	6+	2+	78	7	3					7	↕	↕	↕	↕
ASR	ASRA													47	2	1	A, B, M: b7 → b7 … b0 → C	7	↕	↕	●	↕
	ASRB													57	2	1		7	↕	↕	●	↕
	ASR				07	6	2	67	6+	2+	77	7	3					7	↕	↕	●	↕
BIT	BITA	85	2	2	95	4	2	A5	4+	2+	B5	5	3			1	Bit Test A(M∧A)	●	↕	↕	0	●
	BITB	C5	2	2	D5	4	2	E5	4+	2+	F5	5	3			1	Bit Test B(M∧B)	●	↕	↕	0	●
CLR	CLRA													4F	2	1	0→A	●	0	1	0	0
	CLRB													5F	2	1	0→B	●	0	1	0	0
	CLR				0F	6	2	6F	6+	2+	7F	7	3				0→M	●	0	1	0	0
CMP	CMPA	81	2	2	91	4	2	A1	4+	2+	B1	5	3				Compare M from A	7	↕	↕	↕	↕
	CMPB	C1	2	2	D1	4	2	E1	4+	2+	F1	5	3				Compare M from B	7	↕	↕	↕	↕
	CMPD	10	5	4	10	7	3	10	7+	3+	10	8	4				Compare M:M+1 from D	●	↕	↕	↕	↕
		83			93			A3			B3											
	CMPS	11	5	4	11	7	3	11	7+	3+	11	8	4				Compare M:M+1 from S	●	↕	↕	↕	↕
		8C			9C			AC			BC											
	CMPU	11	5	4	11	7	3	11	7+	3+	11	8	4				Compare M:M+1 from U	●	↕	↕	↕	↕
		83			93			A3			B3											
	CMPX	8C	4	3	9C	6	2	AC	6+	2+	BC	7	3				Compare M:M+1 from X	●	↕	↕	↕	↕
	CMPY	10	5	4	10	7	3	10	7+	3+	10	8	4				Compare M:M+1 from Y	●	↕	↕	↕	↕
		8C			9C			AC			BC											
COM	COMA													43	2	1	$\overline{A}$→A	●	↕	↕	0	1
	COMB													53	2	1	$\overline{B}$→B	●	↕	↕	0	1
	COM				03	6	2	63	6+	2+	73	7	3				$\overline{M}$→M	●	↕	↕	0	1

インストラクション	ニーモニック	イミデエイト Op	~	#	ダイレクト Op	~	#	インデックス[1] Op	~	#	エクステンド Op	~	#	インヘレント Op	~	#	動作	5 H	3 N	2 Z	1 V	0 C
CWAI		3C	≥20	2													CC∧IMM→CC Wait for Interrupt					6
DAA														19	2	1	Decimal Adjust A	●	↕	↕	7	↕
DEC	DECA													4A	2	1	A−1→A	●	↕	↕	↕	●
	DECB													5A	2	1	B−1→B	●	↕	↕	↕	●
	DEC				0A	6	2	6A	6+	2+	7A	7	3				M−1→M	●	↕	↕	↕	●
EOR	EORA	88	2	2	98	4	2	A8	4+	2+	B8	5	3				A⊻M→A	●	↕	↕	0	●
	EORB	C8	2	2	D8	4	2	E8	4+	2+	F8	5	3				B⊻M→B	●	↕	↕	0	●
EXG	R1, R2	1E	8	2													R1↔R2[2]	●	●	●	●	●
INC	INCA													4C	2	1	A+1→A	●	↕	↕	↕	●
	INCB													5C	2	1	B+1→B	●	↕	↕	↕	●
	INC				0C	6	2	6C	6+	2+	7C	7	3				M+1→M	●	↕	↕	↕	●
JMP					0E	3	2	6E	3+	2+	7E	4	3				EA[3]→PC	●	●	●	●	●
JSR					9D	7	2	AD	7+	2+	BD	8	3				Jump to Subroutine	●	●	●	●	●
LD	LDA	86	2	2	96	4	2	A6	4+	2+	B6	5	3				M→A	●	↕	↕	0	●
	LDB	C6	2	2	D6	4	2	E6	4+	2+	F6	5	3				M→B	●	↕	↕	0	●
	LDD	CC	3	3	DC	5	2	EC	5+	2+	FC	6	3				M:M+1→D	●	↕	↕	0	●
	LDS	10	4	4	10	6	3	10	6+	3+	10	7	4				M:M+1→S	●	↕	↕	0	●
		CE			DE			EE			FE											
	LDU	CE	3	3	DE	5	2	EE	5+	2+	FE	6	3				M:M+1→U	●	↕	↕	0	●
	LDX	8E	3	3	9E	5	2	AE	5+	2+	BE	6	3				M:M+1→X	●	↕	↕	0	●
	LDY	10	4	4	10	6	3	10	6+	3+	10	7	4				M:M+1→Y	●	↕	↕	0	●
		8E			9E			AE			BE											
LEA	LEAS							32	4+	2+							EA[3]→S	●	●	●	●	●
	LEAU							33	4+	2+							EA[3]→U	●	●	●	●	●
	LEAX							30	4+	2+							EA[3]→X	●	●	↕	●	●
	LEAY							31	4+	2+							EA[3]→Y	●	●	↕	●	●
LSL	LSLA													48	2	1	A, B, M: c ← b7 … b0 ← 0	7	↕	↕	↕	↕
	LSLB													58	2	1		7	↕	↕	↕	↕
	LSL				08	6	2	68	6+	2+	78	7	3					7	↕	↕	↕	↕
LSR	LSRA													44	2	1	A, B, M: 0 → b7 … b0 → c	●	0	↕	●	↕
	LSRB													54	2	1		●	0	↕	●	↕
	LSR				04	6	2	64	6+	2+	74	7	3					●	0	↕	●	↕
MUL														3D	11	1	A×B→D (Unsigned)	●	●	↕	●	8

インストラクション	ニーモニック	アドレッシングモード															動　作	5	3	2	1	0
		イミデエイト			ダイレクト			インデックス[1]			エクステンド			インヘレント								
		Op	~	#	Op	~	#	Op	~	#	Op	~	#	Op	~	#		H	N	Z	V	C
NEG	NEGA													40	2	1	$\overline{A}$+1→A	7	↕	↕	↕	↕
	NEGB													50	2	1	$\overline{B}$+1→B	7	↕	↕	↕	↕
	NEG				00	6	2	60	6+	2+	70	7	3				$\overline{M}$+1→M	7	↕	↕	↕	↕
NOP														12	2	1	No Operation	●	●	●	●	●
OR	ORA	8A	2	2	9A	4	2	AA	4+	2+	BA	5	3				A V M→A	●	↕	↕	0	●
	ORB	CA	2	2	DA	4	2	EA	4+	2+	FA	5	3				B V M→B	●	↕	↕	0	●
	ORCC	1A	3	2													CC V IMM→CC				6	
PSH	PSHS	34	5+[4]	2													Push Registers on S Stack	●	●	●	●	●
	PSHU	36	5+[4]	2													Push Registers on U Stack	●	●	●	●	●
PUL	PULS	35	5+[4]	2													Pull Registers from S Stack	●	●	●	●	●
	PULU	37	5+[4]	2													Pull Registers from U Stack	●	●	●	●	●
ROL	ROLA													49	2	1	A	●	↕	↕	↕	↕
	ROLB													59	2	1	B	●	↕	↕	↕	↕
	ROL				09	6	2	69	6+	2+	79	7	3				M　c　b7　b0	●	↕	↕	↕	↕
ROR	RORA													46	2	1	A	●	↕	↕	●	↕
	RORB													56	2	1	B	●	↕	↕	●	↕
	ROR				06	6	2	66	6+	2+	76	7	3				M　c　b7　b0	●	↕	↕	●	↕
RTI														3B	6/15	1	Return from Interrupt					6
RTS														39	5	1	Return from Subroutine	●	●	●	●	●
SBC	SBCA	82	2	2	92	4	2	A2	4+	2+	B2	5	3				A-M-C→A	7	↕	↕	↕	↕
	SBCB	C2	2	2	D2	4	2	E2	4+	2+	F2	5	3				B-M-C→B	7	↕	↕	↕	↕
SEX														1D	2	1	Sign Extend B into A	●	↕	↕	0	●
ST	STA				97	4	2	A7	4+	2+	B7	5	3				A→M	●	↕	↕	0	●
	STB				D7	4	2	E7	4+	2+	F7	5	3				B→M	●	↕	↕	0	●
	STD				DD	5	2	ED	5+	2+	FD	6	3				D→M:M+1	●	↕	↕	0	●
	STS				10	6	3	10	6+	3+	10	7	4				S→M:M+1	●	↕	↕	0	●
					DF			EF			FF											
	STU				DF	5	2	EF	5+	2+	FF	6	3				U→M:M+1	●	↕	↕	0	●
	STX				9F	5	2	AF	5+	2+	BF	6	3				X→M:M+1	●	↕	↕	0	●
	STY				10	6	3	10			10	7	4				Y→M:M+1	●	↕	↕	0	●
					9F			AF	6+	3+	BF											

インストラクション	ニーモニック	アドレッシングモード															動作	5	3	2	1	0
		イミデエイト			ダイレクト			インデックス[1]			エクステンド			インヘレント								
		Op	~	#	Op	~	#	Op	~	#	Op	~	#	Op	~	#		H	N	Z	V	C
SUB	SUBA	80	2	2	90	4	2	A0	4+	2+	B0	5	3				A-M→A	7	↕	↕	↕	↕
	SUBB	C0	2	2	D0	4	2	E0	4+	2+	F0	5	3				B-M→B	7	↕	↕	↕	↕
	SUBD	83	4	3	93	6	2	A3	6+	2+	B3	7	3				D-M:M+1→D	●	↕	↕	↕	↕
SWI	SWI[5]													3F	19	1	Software Interrupt 1	●	●	●	●	●
	SWI2[5]													10 3F	20	2	Software Interrupt 2	●	●	●	●	●
	SWI3[5]													11 3F	20	1	Software Interrupt 3	●	●	●	●	●
SYNC														13	≧4	1	Synchronize to Interrupt	●	●	●	●	●
TFR	R1, R2													1F	6	2	R1→R2[2]	●	●	●	●	●
TST	TSTA													4D	2	1	Test A	●	↕	↕	0	●
	TSTB													5D	2	1	Test B	●	↕	↕	0	●
	TST				0D	6	2	6D	6+	2+	7D	7	3				Test M	●	↕	↕	0	●

Op　オペレーションコード(16進数)
~　MPUの実行サイクル数
#　命令語のバイト数
+　算術加算
-　算術減算
●　算術乗算

$\overline{M}$　Mの補数
→　転送方向
↕　命令実行の結果に応じて変化
●　その命令では変化しない
CC　コンディションコードレジスタ
:　比較
∨　論理和
∧　論理積
∀　排他的論理和

E　エンタイアフラグ
F　FIRQマスクフラグ
$H(b_5)$　b_3からのハーフキャリー
I　IRQマスクフラグ
$N(b_3)$　負のフラグ
$Z(b_2)$　ゼロのフラグ
$V(b_1)$　2の補数のオーバフローフラグ
$C(b_0)$　b_7からのキャリー/b_7へのボロー

ブランチ命令表

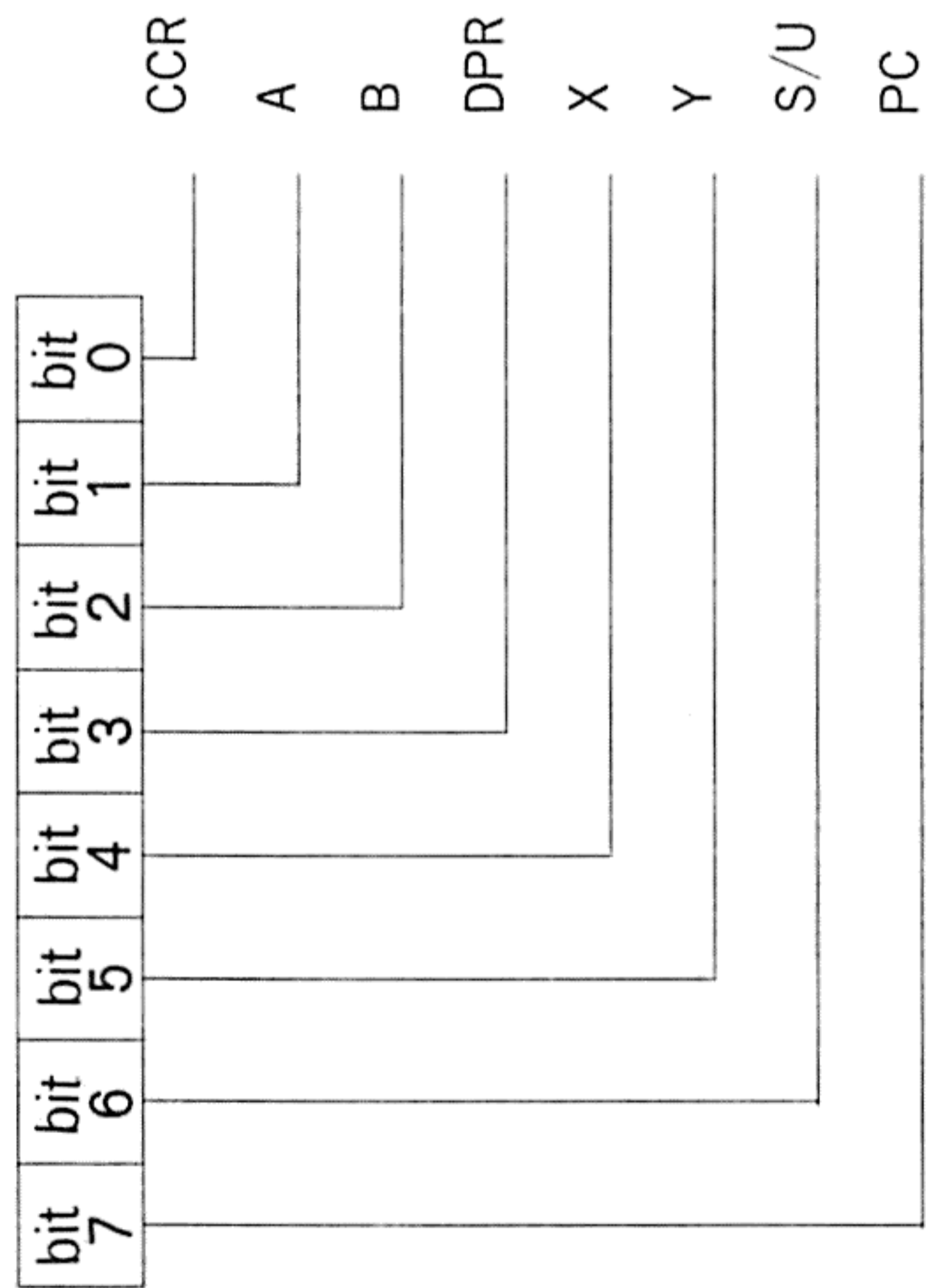
インストラクション	ニーモニック	Op	～#	#	動作作	5 H	3 N	2 Z	1 V	0 C
BCC	BCC	24	3	2	Branch C=0	●	●	●	●	●
	LBCC	10	5(6)	4	Long Branch	●	●	●	●	●
		24			C=0					
BCS	BCS	25	3	2	Branch C=1	●	●	●	●	●
	LBCS	10	5(6)	4	Long Branch	●	●	●	●	●
		25			C=1					
BEQ	BEQ	27	3	2	Branch Z=1	●	●	●	●	●
	LBEQ	10	5(6)	4	Long Branch	●	●	●	●	●
		27			Z=1					
BGE	BGE	2C	3	2	Branch≥Zero	●	●	●	●	●
	LBGE	10	5(6)	4	Long Branch≥Zero	●	●	●	●	●
		2C								
BGT	BGT	2E	3	2	Branch>Zero	●	●	●	●	●
	LBGT	10	5(6)	4	Long Branch>Zero	●	●	●	●	●
		2E								
BHI	BHI	22	3	2	Branch Higher	●	●	●	●	●
	LBHI	10	5(6)	4	Long Branch Higher	●	●	●	●	●
		22								
BHS	BHS	24	3	2	Branch Higher or Same	●	●	●	●	●
	LBHS	10	5(6)	4	Long Branch Higher or Same	●	●	●	●	●
		24								
BLE	BLE	2F	3	2	Branch≤Zero	●	●	●	●	●
	LBLE	10	5(6)	4	Long Branch≤Zero	●	●	●	●	●
		2F								
BLO	BLO	25	3	2	Branch lower	●	●	●	●	●
	LBLO	10	5(6)	4	Long Branch Lower	●	●	●	●	●
		25								

インストラクション	ニーモニック	アドレッシングモード リラティブ Op	～	##	動　作	5 H	3 N	2 Z	1 V	0 C
BLS	BLS	23	3	2	Branch Lower or Same	●	●	●	●	●
	LBLS	10	5(6)	4	Long Branch Lower or Same	●	●	●	●	●
		23								
BLT	BLT	2D	3	2	Branch<Zero	●	●	●	●	●
	LBLT	10	5(6)	4	Long Branch<Zero	●	●	●	●	●
		2D								
BMI	BMI	2B	3	2	Branch Minus	●	●	●	●	●
	LBMI	10	5(6)	4	Long Branch Minus	●	●	●	●	●
		2B								
BNE	BNE	26	3	2	Branch Z=0	●	●	●	●	●
	LBNE	10	5(6)	4	Long Branch	●	●	●	●	●
		26			Z=0					
BPL	BPL	2A	2	2	Branch Plus	●	●	●	●	●
	LBPL	10	5(6)	4	Long Branch Plus	●	●	●	●	●
		2A								
BRA	BRA	20	3	2	Branch Ahways	●	●	●	●	●
	LBRA	16	5	4	Long Branch Always	●	●	●	●	●
BRN	BRN	21	3	2	Branch Never	●	●	●	●	●
	LBRN	10	5	4	Long Branch Never	●	●	●	●	●
		21								
BSR	BSR	8D	7	2	Branch to Subroutine	●	●	●	●	●
	LBSR	17	9	3	Long Branch to Subroutine	●	●	●	●	●
BVC	BVC	28	3	2	Branch V=0	●	●	●	●	●
	LBVC	10	5(6)	4	Long Branch	●	●	●	●	●
		28			V=0					
BVS	BVS	29	3	2	Branch V=1	●	●	●	●	●
	LBVS	10	5(6)	4	Long Branch	●	●	●	●	●
		29			V=1					

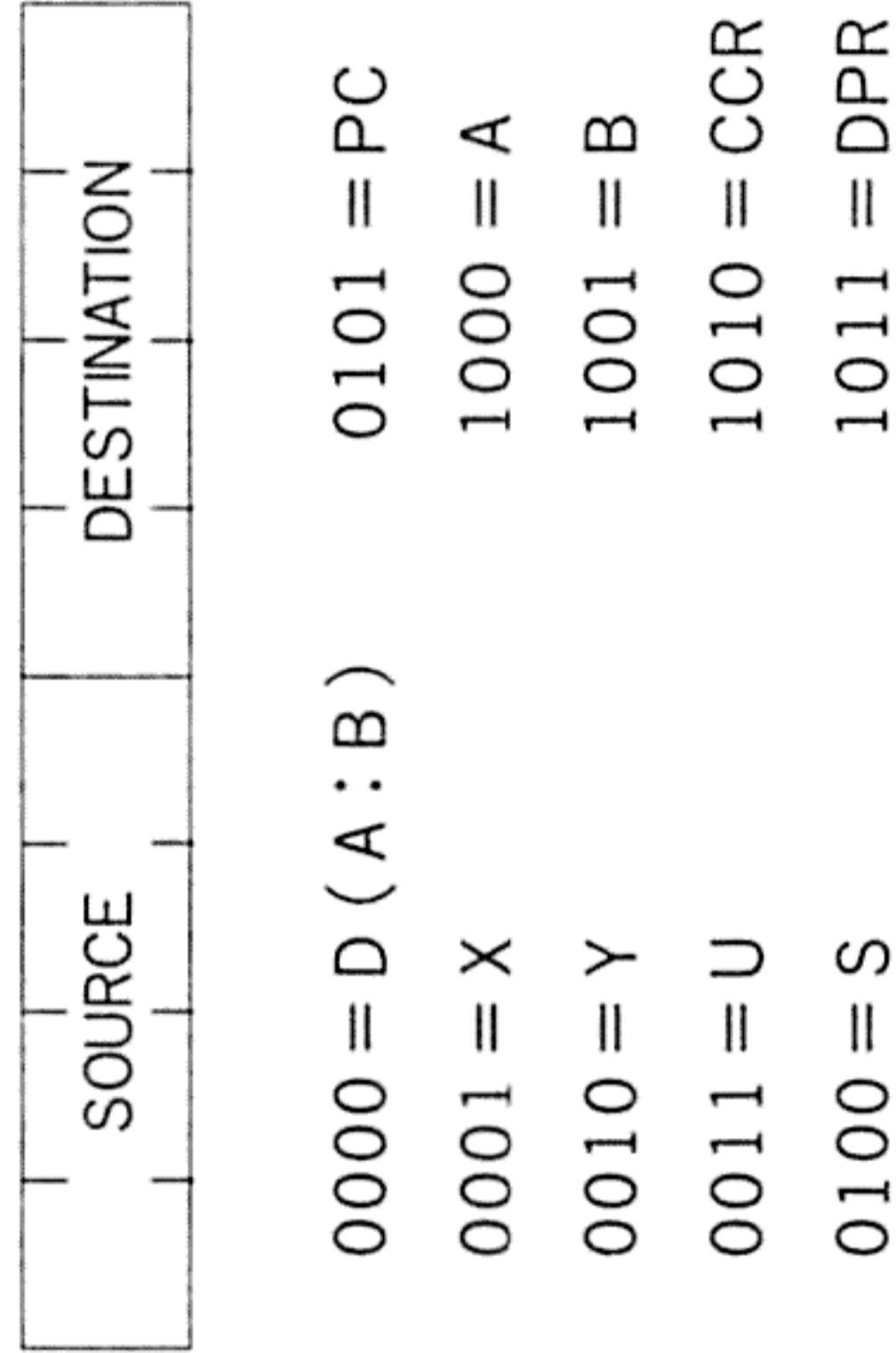
注：

1．インデックスアドレッシングモードの時は，この欄に書かれているバイト数，実行サイクル数に,「インデックスアドレッシングモードのポストバイト」の表にある値を加えたものが実際の値となります。
2．Ｒ１とＲ２は、同じバイト数のレジスタでなければいけません。
　8ビットレジスタ：A，B，CC，DP
　16ビットレジスタ：X，Y，U，S，D，PC
3．ＥＡは，実効アドレス（Effective address）を示します。
4．PSH，PUL命令は，この値に，プッシュ又はプルするバイト数を加えた値となります。
5．ＳＷＩはＩフラグとＦフラグを１にしますが，SWI２，SWI３は，Ｉフラグ，Ｆフラグには影響しません。
6．ＣＣレジスタは，命令の結果によってセット又はリセットされます。
7．未定義
8．MUL命令の時に限り，Ｃフラグはbit７と同じ値をとります。

PUSH/PULLのポストバイト

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
PC	S/U	Y	X	DPR	B	A	CCR

TRANSFER/EXCHANGEのポストバイト

SOURCE	DESTINATION

0000＝D（A：B）	0101＝PC
0001＝X	1000＝A
0010＝Y	1001＝B
0011＝U	1010＝CCR
0100＝S	1011＝DPR

インデックスアドレッシングモードのポストバイト

アドレッシング モード			インダイレクトでない場合				インダイレクトの場合			
			アセンブラ形式	ポスト バイト	± ~	± #	アセンブラ形式	ポスト バイト	± ~	± #
インデックスト	コンスタント オフセット	オフセットなし	，R	1RR00100	0	0	〔，R〕	1RR10100	3	0
		5ビット オフセット	n，R	0RRnnnnn	1	0	，—	—	—	—
		8ビット オフセット	n，R	1RR01000	1	1	〔n，R〕	1RR11000	4	1
		16ビット オフセット	n，R	1RR01001	4	2	〔n，R〕	1RR11001	7	2
	アキュムレータ オフセット	Aレジスタ オフセット	A，R	1RR00110	1	0	〔A，R〕	1RR10110	4	0
		Bレジスタ オフセット	B，R	1RR00101	1	0	〔B，R〕	1RR10101	4	0
		Dレジスタ オフセット	D R	1RR01011	4	0	〔D，R〕	1RR11011	7	0
	オート インクリメント／デクリメント	インクリメント(＋1)	，R＋	1RR00000	2	0	—	—	—	—
		インクリメント(＋2)	，R＋＋	1RR00001	3	0	〔，R＋＋〕	1RR10001	6	0
		デクリメント (－1)	，－R	1RR00010	2	0	—	—	—	—
		デクリメント (－2)	，－－R	1RR00011	3	0	〔，－－R〕	1RR10011	6	0
プログラムカウンタ リラティブ		8ビット オフセット	n，PCR	1XX01100	1	1	〔n，PCR〕	1XX11100	4	1
		16ビット オフセット	n，PCR	1XX01101	5	2	〔n，PCR〕	1XX11101	8	2
エクステンディッド インダイレクト		16ビット アドレス	—	—	—	—	〔n〕	10011111	5	2

〈記号〉 R：X RR：00＝X X：Don't care ±＝追加されるマシンサイクル数

Y 01＝Y ±#＝追加されるバイト数

U 10＝U

S 11＝S

ブランチ

	OP	~	#
BRA	20	3	2
LBRA	16	5	3
BRN	21	3	2
LBRN	1021	5	4
BSR	8D	7	2
LBSR	17	9	3

符号付きブランチ

Test	True	OP	False	OP
r>m	BGT	2E	BLE	2F
r≧m	BGE	2C	BLT	2D
r=m	BEQ	27	BNE	26
r≦m	BLE	2F	BGT	2E
r<m	BLT	2D	BGE	2C

条件付きブランチ

Test	True	OP	False	OP
N=1	BMI	2B	BPL	2A
Z=1	BEQ	27	BNE	26
V=1	BVS	29	BVC	28
C=1	BCS	25	BCC	24

符号なしブランチ

Test	True	OP	False	OP
r>m	BHI	22	BLS	23
r≧m	BHS	24	BLO	25
r=m	BEQ	27	BNE	26
r≦m	BLS	23	BHI	22
r<m	BLO	25	BHS	24

スタッフ格納順序

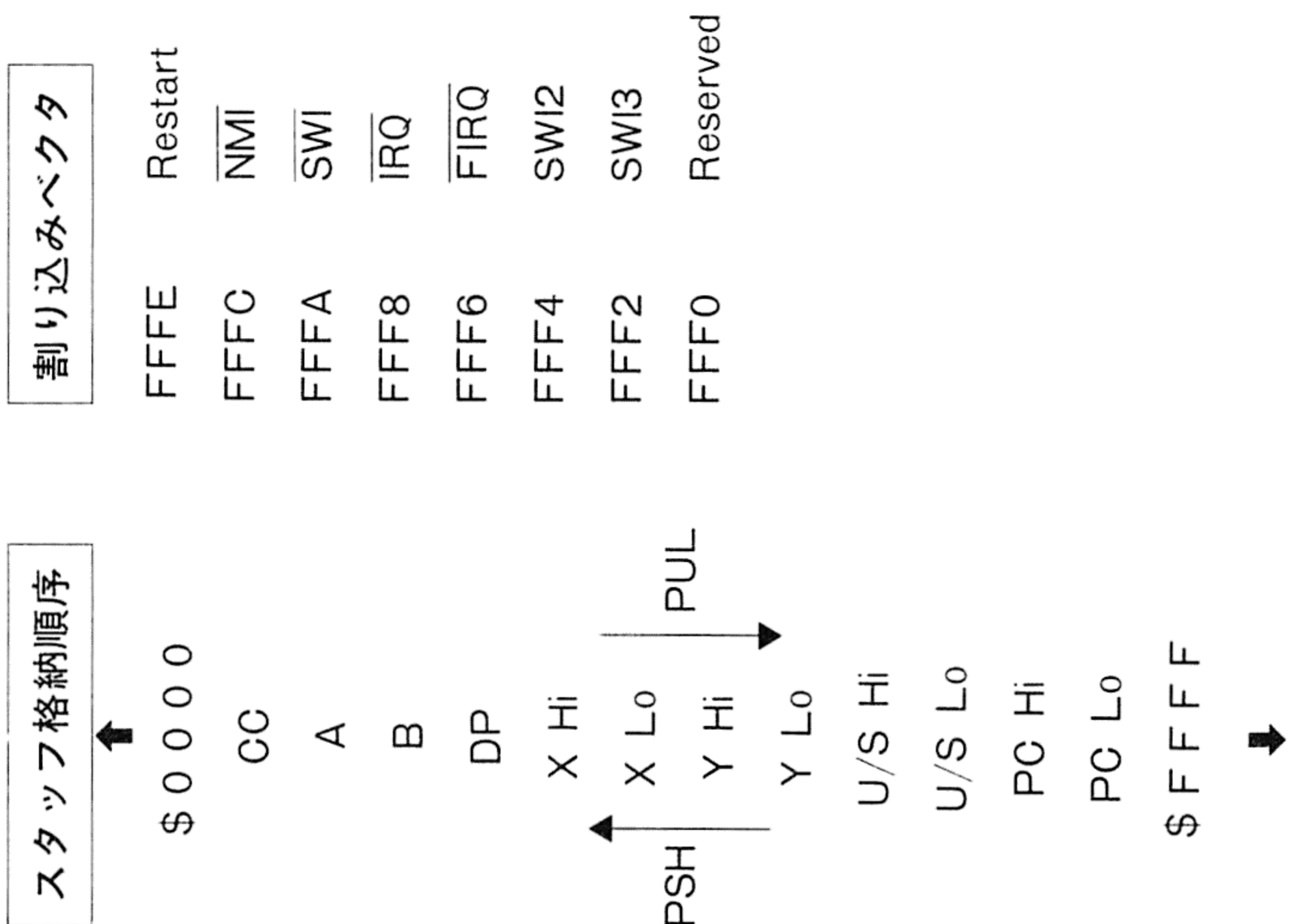
↑

$ 0 0 0 0
CC
A
B
DP
X Hi
X Lo
Y Hi
Y Lo
U/S Hi
U/S Lo
PC Hi
PC Lo
$ F F F F

↓

PSH (↑)　PUL (↓)

割り込みベクタ

FFFE	Restart
FFFC	$\overline{\text{NMI}}$
FFFA	$\overline{\text{SWI}}$
FFF8	$\overline{\text{IRQ}}$
FFF6	$\overline{\text{FIRQ}}$
FFF4	SWI2
FFF2	SWI3
FFF0	Reserved

OP	ニーモニック	アドレッシングモード	~	#	OP	ニーモニック	アドレッシングモード	~	#	OP	ニーモニック	アドレッシングモード	~	#
00	NEG	DIRECT	6	2	1C	ANDCC	IMMED	3	2	2E	BGT	RELATIVE	3	2
03	COM	↑	6	2	1D	SEX	INHERENT	2	1	2F	BLE	RELATIVE	3	2
04	LSR	│	6	2	1E	EXG	IMMED	8	2	30	LEAX	INDEXED	4	2
06	ROR	│	6	2	1F	TFR	INHERENT	6	2	31	LEAY	↑	4	2
07	ASR	│	6	2	20	BRA	RELATIVE	3	2	32	LEAS	↓	4	2
08	ASL/LSL	│	6	2	21	BRN	↑	3	2	33	LEAU	INDEXED	4	2
09	ROL	│	6	2	22	BHI	│	3	2	34	PSHS	IMMED	5	2
0A	DEC	│	6	2	23	BLS	│	3	2	35	PULS	↑	5	2
0C	INC	│	6	2	24	BHS/BCC	│	3	2	36	PSHU	↓	5	2
0D	TST	│	6	2	25	BLO/BCS	│	3	2	37	PULU	IMMED	5	2
0E	JMP	↓	3	2	26	BNE	│	3	2	39	RTS	INHERENT	5	1
0F	CLR	DIRECT	6	2	27	BEQ	│	3	2	3A	ABX	↕	3	1
12	NOP	INHERENT	2	1	28	BVC	│	3	2	3B	RTI	INHERENT	6/15	1
13	SYNC	INHERENT	4	1	29	BVS	│	3	2	3C	CWAI	IMMED	20	2
16	LBRA	RELATIVE	5	3	2A	BPL	│	3	2	3D	MUL	INHERENT	11	1
17	LBSR	RELATIVE	9	3	2B	BMI	│	3	2	3F	SWI	↑	19	1
19	DAA	INHERENT	2	1	2C	BGE	↓	3	2	40	NEGA	↓	2	1
1A	ORCC	IMMED	3	2	2D	BLT	RELATIVE	3	2	43	COMA	INHERENT	2	1

OP	ニーモニック	アドレッシングモード	~	#	OP	ニーモニック	アドレッシングモード	~	#	OP	ニーモニック	アドレッシングモード	~	#
44	LSRA	INHERENT	2	1	5D	TSTB	INHERENT	2	1	77	ASR	EXTENDED	7	3
46	RORA		2	1	5F	CLRB	INHERENT	2	1	78	ASL/LSL		7	3
47	ASRA		2	1	60	NEG	INDEXED	6	2	79	ROL		7	3
48	ASLA/LSLA		2	1	63	COM		6	2	7A	DEC		7	3
49	ROLA		2	1	64	LSR		6	2	7C	INC		7	3
4A	DECA		2	1	66	ROR		6	2	7D	TST		7	3
4C	INCA		2	1	67	ASR		6	2	7E	JMP		4	3
4D	TSTA		2	1	68	ASL/LSL		6	2	7F	CLR	EXTENDED	7	3
4F	CLRA		2	1	69	ROL		6	2	80	SUBA	IMMED	2	2
50	NEGB		2	1	6A	DEC		6	2	81	CMPA		2	2
53	COMB		2	1	6C	INC		6	2	82	SBCA		2	2
54	LSRB		2	1	6D	TST		6	2	83	SUBD		4	3
56	RORB		2	1	6E	JMP		3	2	84	ANDA		2	2
57	ASRB		2	1	6F	CLR	INDEXED	6	2	85	BITA		2	2
58	ASLB/LSLB		2	1	70	NEG	EXTENDED	7	3	86	LDA		2	2
59	ROLB		2	1	73	COM		7	3	88	EORA		2	2
5A	DECB		2	1	74	LSR		7	3	89	ADCA		2	2
5C	INCB	INHERENT	2	1	76	ROR	EXTENDED	7	3	8A	ORA	IMMED	2	2

OP	ニーモニック	アドレッシングモード	~	#	OP	ニーモニック	アドレッシングモード	~	#	OP	ニーモニック	アドレッシングモード	~	#
8B	ADDA	IMMED	2	2	9E	LDX	DIRECT	5	2	B0	SUBA	EXTENDED	5	3
8C	CMPX	IMMED	4	3	9F	STX	DIRECT	5	2	B1	CMPA		5	3
8D	BSR	RELATIVE	7	2	A0	SUBA	INDEXED	4	2	B2	SBCA		5	3
8E	LDX	IMMED	3	3	A1	CMPA		4	2	B3	SUBD		7	3
90	SUBA	DIRECT	4	2	A2	SBCA		4	2	B4	ANDA		5	3
91	CMPA		4	2	A3	SUBD		6	2	B5	BITA		5	3
92	SBCA		4	2	A4	ANDA		4	2	B6	LDA		5	3
93	SUBD		6	2	A5	BITA		4	2	B7	STA		5	3
94	ANDA		4	2	A6	LDA		4	2	B8	EORA		5	3
95	BITA		4	2	A7	STA		4	2	B9	ADCA		5	3
96	LDA		4	2	A8	EORA		4	2	BA	ORA		5	3
97	STA		4	2	A9	ADCA		4	2	BB	ADDA		5	3
98	EORA		4	2	AA	ORA		4	2	BC	CMPX		7	3
99	ADCA		4	2	AB	ADDA		4	2	BD	JSR		8	3
9A	ORA		4	2	AC	CMPX		6	2	BE	LDX		6	3
9B	ADDA		4	2	AD	JSR		7	2	BF	STX	EXTENDED	6	3
9C	CMPX		6	2	AE	LDX		5	2	C0	SUBB	IMMED	2	2
9D	JSR	DIRECT	7	2	AF	STX	INDEXED	5	2	C1	CMPB	IMMED	2	2

OP	ニーモモック	アドレッシングモード	~	#	OP	ニーモニック	アドレッシングモード	~	#	OP	ニーモニック	アドレッシングモード	~	#
C2	SBCB	IMMED	2	2	D7	STB	DIRECT	4	2	E9	ADCB	INDEXED	4	2
C3	ADDD	↑	4	3	D8	EORB	↑	4	2	EA	ORB	↑	4	2
C4	ANDB	│	2	2	D9	ADCB	│	4	2	EB	ADDB	│	4	2
C5	BITB	│	2	2	DA	ORB	│	4	2	EC	LDD	│	5	2
C6	LDB	│	2	2	DB	ADDB	│	4	2	ED	STD	│	5	2
C8	EORB	│	2	2	DC	LDD	│	5	2	EE	LDU	↓	5	2
C9	ADCB	│	2	2	DD	STD	│	5	2	EF	STU	INDEXED	5	2
CA	ORB	│	2	2	DE	LDU	↓	5	2	F0	SUBB	EXTENDED	5	3
CB	ADDB	│	2	2	DF	STU	DIRECT	5	2	F1	CMPB	↑	5	3
CC	LDD	↓	3	3	E0	SUBB	INDEXED	4	2	F2	SBCB	│	5	3
CE	LDU	IMMED	3	3	E1	CMPB	↑	4	2	F3	ADDD	│	7	3
D0	SUBB	DIRECT	4	2	E2	SBCB	│	4	2	F4	ANDB	│	5	3
D1	CMPB	↑	4	2	E3	ADDD	│	6	2	F5	BITB	│	5	3
D2	SBCB	│	4	2	E4	ANDB	│	4	2	F6	LDB	│	5	3
D3	ADDD	│	6	2	E5	BITB	│	4	2	F7	STB	│	5	3
D4	ANDB	│	4	2	E6	LDB	│	4	2	F8	EORB	│	5	3
D5	BITB	↓	4	2	E7	STB	↓	4	2	F9	ADCB	↓	5	3
D6	LDB	DIRECT	4	2	E8	EORB	INDEXED	4	2	FA	ORB	EXTENDED	5	3

OP	ニーモニック	アドレッシングモード	~	#	OP	ニーモニック	アドレッシングモード	~	#	OP	ニーモニック	アドレッシングモード	~	#
FB	ADDB	EXTENDED	5	3	102E	LBGT	RELATIVE	5(6)	4	10CE	LDS	IMMED	4	4
FC	LDD		6	3	102F	LBLE	RELATIVE	5(6)	4	10DE	LDS	DIRECT	6	3
FD	STD		6	3	103F	SWI2	INHERENT	20	2	10DF	STS	DIRECT	6	3
FE	LDU		6	3	1083	CMPD	IMMED	5	4	10EE	LDS	INDEXED	6	3
FF	STU	EXTENDED	6	3	108C	CMPY		5	4	10EF	STS	INDEXED	6	3
1021	LBRN	RELATIVE	5	4	108E	LDY	IMMED	4	4	10FE	LDS	EXTENDED	7	4
1022	LBHI		5(6)	4	1093	CMPD	DIRECT	7	3	10FF	STS	EXTENDED	7	4
1023	LBLS		5(6)	4	109C	CMPY		7	3	113F	SWI3	INHERENT	20	2
1024	LBHS/LBCC		5(6)	4	109E	LDY		6	3	1183	CMPU	IMMED	5	4
1025	LBCS/LBLO		5(6)	4	109F	STY	DIRECT	6	3	118C	CMPS	IMMED	5	4
1026	LBNE		5(6)	4	10A3	CMPD	INDEXED	7	3	1193	CMPU	DIRECT	7	3
1027	LBEQ		5(6)	4	10AC	CMPY		7	3	119C	CMPS	DIREECT	7	3
1028	LBVC		5(6)	4	10AE	LDY		6	3	11A3	CMPU	INDEXED	7	3
1029	LBVS		5(6)	4	10AF	STY	INDEXED	6	3	11AC	CMPS	INDEXED	7	3
102A	LBPL		5(6)	4	10B3	CMPD	EXTENDED	8	4	11B3	CMPU	EXTENDED	8	4
102B	LBMI		5(6)	4	10BC	CMPY		8	4	11BC	CMPS	EXTENDED	8	4
102C	LBGE		5(6)	4	10BE	LDY		7	4					
102D	LBLT	RELATIVE	5(6)	4	10BF	STY	EXTENDED	7	4					

著　者　中村 英都

発行者　牧谷秀昭
発行所　秀和システムトレーディング株式会社
郵便番号１０７　東京都港区南青山２－１９－５　関原ビル

SHUWA SYSTEM TRADING CO., LTD.
SEKIHARA BLDG,
2-19-5 MINAMIAOYAMA, MINATO-KU, TOKYO, 107 JAPAN

印刷所　東京スガキ印刷株式会社

１９８３年１１月１日　第１刷発行　１９８４年３月２４日　第２刷発行